Interpretable Artificial Intelligence: A Comparative Study of LLM Explanation Methods

---

Ruslan Sidukov

Matriculation number: 423443

September 15, 2026

Rheinland-Pfälzische Technische Universität
Kaiserslautern
Landau

Department of Computer Science

Bachelor Thesis

Interpretable Artificial Intelligence: A Comparative Study of LLM Explanation Methods

Ruslan Sidukov

Matriculation number: 423443

1. Reviewer apl. Prof. Dr. Achim Ebert
Department of Computer Science
Human-Computer Interaction Lab

2. Reviewer Kai Klingshirn, M.Sc.
Department of Computer Science
Human-Computer Interaction Lab

Submission date: September 15, 2026

Ruslan Sidukov

Interpretable Artificial Intelligence: A Comparative Study of LLM Explanation Methods

Bachelor Thesis, September 15, 2026

Reviewers: apl. Prof. Dr. Achim Ebert and Kai Klingshirn, M.Sc.

Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau (RPTU)

Gottlieb-Daimler-Straße 47

67663 Kaiserslautern

Department of Computer Science

Abstract

Large language models are increasingly used in areas where a wrong answer may cause real harm, and a user who wants to know why a model gave a particular answer must rely on an explanation method. Several families of such methods exist, but they are rarely compared against one another on the same terms. This matters because a user is shown the output of one method and has no way of knowing whether another would have pointed somewhere else. This thesis compares three families - attention weights, Integrated Gradients and leave-one-out - scoring all three with the same faithfulness and robustness metrics. However, no method wins on every criterion: leave-one-out scores highest on one faithfulness metric, attention on the second faithfulness metric and on robustness, and Integrated Gradients leads on none, even though it has the strongest theoretical foundation of the three. The methods also disagree almost completely, which means two of them looking at the same answer mostly point at different words. Which method to trust depends on which property matters for a given case. Since the best method cannot be chosen by measurement, this thesis also treats the choice as a design problem. A browser-based prototype was developed that computes all three explanations for one answer and lets a user move between them. Moreover, a user study that would test whether this helps is specified, but it was not conducted, so the interface is presented as a design rather than as a measured result.

Zusammenfassung

Große Sprachmodelle werden zunehmend in Bereichen eingesetzt, in denen eine falsche Antwort realen Schaden verursachen kann, und wer wissen möchte, warum ein Modell eine bestimmte Antwort gegeben hat, ist auf ein Erklärungsverfahren angewiesen. Es gibt mehrere Familien solcher Verfahren, sie werden jedoch selten unter denselben Bedingungen miteinander verglichen. Das ist von Bedeutung, weil jeweils die Ausgabe eines Verfahrens gezeigt wird, ohne dass erkennbar ist, ob ein anderes auf andere Wörter gezeigt hätte. Diese Arbeit vergleicht drei Familien – Aufmerksamkeitsgewichte (*attention weights*), Integrated Gradients und Leave-one-out – und bewertet alle drei mit denselben Metriken für Treue (*faithfulness*) und Robustheit. Dabei gewinnt kein Verfahren in allen Kriterien: Leave-one-out erreicht den höchsten Wert bei der einen Treue-Metrik, Aufmerksamkeitsgewichte bei der zweiten Treue-Metrik und bei der Robustheit, und Integrated Gradients führt in keinem, obwohl es von den dreien theoretisch am besten fundiert ist. Die Verfahren widersprechen einander zudem fast vollständig, was bedeutet, dass zwei von ihnen, die dieselbe Antwort betrachten, überwiegend auf unterschiedliche Wörter zeigen. Welchem Verfahren zu trauen ist, hängt davon ab, welche Eigenschaft im jeweiligen Fall wichtig ist. Da sich das beste Verfahren nicht durch Messung bestimmen lässt, behandelt diese Arbeit die Wahl zugleich als Entwurfsproblem. Es wurde ein browserbasierter Prototyp entwickelt, der alle drei Erklärungen zu einer Antwort berechnet und es ermöglicht, zwischen ihnen zu wechseln. Darüber hinaus wird eine Nutzerstudie spezifiziert, die prüfen würde, ob dies hilft. Sie wurde jedoch nicht durchgeführt, sodass die Oberfläche als Entwurf und nicht als gemessenes Ergebnis dargestellt wird.

Contents

1 Introduction **1**

2 Background: Explainable AI and Large Language Models **5**
2.1 Transformers and Autoregressive Language Models 5
2.2 Explainable Artificial Intelligence (XAI) 8
2.3 Placing the Three Families 10

3 Explanation Methods for Language Models **12**
3.1 Attention-based Explanations 13
3.2 Gradient-based Explanations 18
3.3 Leave-One-Out Explanations 24
3.4 Comparing the Three Families 27

4 Computational Evaluation **30**
4.1 Implementation 30
4.2 Criteria for Evaluating Explanations 33
4.3 Choosing a Faithfulness Metric 34
4.4 Evaluation Framework 35
4.5 Hypotheses 38
4.6 Tasks and Prompts 39
4.7 Results 40

5 User Evaluation **46**
5.1 Requirements for Explanation Interfaces 46
5.2 Existing Interactive XAI Tools 48
5.3 Interface Design 49
5.4 Implementation of the Interface 53
5.5 Proposed User Study 54

6 Discussion and Conclusion **57**
6.1 Limitations 57
6.2 Future Work 58
6.3 Conclusion 59

Bibliography **61**

1 Introduction

Large language models are widely used by many different users. Companies use them to draft text or answer customer questions, and they are reaching domains where a wrong answer is costly, such as healthcare or autonomous driving [19, 16]. The difficulty is that a user cannot tell when the model is wrong. The model gives an answer in fluent language and offers nothing about how it arrived there, so a correct answer and a confident mistake look exactly the same. The obvious response to this is to make the model explain itself. One way is simply to ask: the model writes an account of its own reasoning alongside the answer. But that account is produced by the same process that produced the answer, so it sounds convincing whether or not it is true, and Atanasova et al. [5] show with two faithfulness tests that such explanations often fail to reflect what the model actually did. Asking the model does not solve the checking problem - it just moves it one step back. What can be checked instead is the model itself. In a small model the construction alone is enough: the coefficients of a linear model state what it does, and a reader who understands the equation understands the decision. This property is called *interpretability*. But a model with billions of parameters spread over many layers cannot be read this way. What can be done instead is to reconstruct the reasons for an answer afterwards, with a separate method built for the purpose. This is *explainability*. The two terms are often used interchangeably, and there is no agreed technical definition of either [19]. This thesis keeps them apart, because the distinction matters more than it seems. An interpretable model shows how it reached its answer. An explainability method reconstructs the reasons from the outside, and the reconstruction can be wrong about the model it describes.

This is where the problem becomes real. A convincing but wrong explanation is worse than no explanation at all, because the user trusts it. The usual way to explain a language model is to highlight the words the model paid attention to: so-called attention weights are computed on every forward pass, they are non-negative and sum to one, and a coloured heatmap over the user's own question is easier to trust than a table of numbers. The intuition behind it is that a word the model attended to strongly is a word that mattered. That intuition does not survive examination. Jain and Wallace [14] showed that a model's attention can be replaced with a completely different distribution while the prediction stays the same. That means that the original attention map cannot be the reason for a model's answer. Wiegreffe

and Pinter [38] replied with a paper called *Attention is not not Explanation*. Their objection is procedural. The adversarial distributions were installed by hand into a trained model, which breaks the link between the weights and the training that produced these weights. A distribution reached this way is, therefore, not the one the model would have used. They also made a theoretical point: attention offers *an* explanation, not *the* explanation. A second account that fits the same prediction does not make the first one false. The discussion has never been settled.

Attention is not the only option. A second family of methods is gradient-based. The model does not work with words directly - it turns each word of the prompt into a list of numbers. Changing one of those lists slightly changes the answer's score by some amount, and that amount is what a gradient reports. A word whose numbers move the score is the one the answer depends on. A word that moves nothing was not doing much. But a gradient only reports what happens under a small change. Consider a model that has already committed to its answer. The score is at its ceiling, so nudging one word does not move it, and the gradient is near zero even for a word the answer depends on completely. The word looks unimportant precisely because it is doing its job so well. The third family is leave-one-out. It is the most direct of the three, since it deletes a word, runs the model again, and measures how far the score of the original answer falls. A large drop means the word mattered. The weakness is that deleting a word creates a sentence that no one would ever write, and the model was never trained on such sentences. So, the drop may also mean the sentence was just broken.

Gradient-based methods and leave-one-out have one important thing in common. Both are computed from the answer: the gradient from the answer's score, the deletion from how much that score drops. Attention, on the other hand, is taken from inside the model as it reads, not from the answer's score. So each method offers something the others do not, and each is doubtful in its own way. Choosing between them is a real decision with real costs.

Anyone wanting to explain a language model therefore has three methods to choose from, they answer three different questions, and they can disagree with one another. Choosing between them requires knowing how far each can be trusted, and the literature does not settle this, because no study compares all three on a model that generates text. The closest is Jain and Wallace [14], the only work reviewed here that measures all three families against each other, but it does so on recurrent classifiers. The others each leave a family out. Serrano and Smith [27] put attention against gradients, and Ali et al. [4] do the same across a range of datasets, but neither deletes a word to see what happens. Bastings and Filippova [6] cover all three and come down in favour of saliency, but they argue the case and do not measure it. Surveys such as Ivanovs et al. [13] catalogue the methods of a single family

without running them against one another. So, the three families have never been compared as a set, and every study named here works on a classifier, a model that reads an input once and produces one score per class. In this setting an explanation has an obvious target: the score for the predicted class. A decoder-only model, however, produces its answer one token at a time, so no single number is waiting to be explained, and what an explanation should even be about has to be decided before anything can be measured.

This thesis addresses both. It applies attention-based, gradient-based, and leave-one-out explanations to one decoder-only language model, and measures them against each other on two criteria: *faithfulness*, which asks whether an explanation reflects what the model actually used, and *robustness*, which asks whether an explanation stays the same when the question is paraphrased without changing its meaning. A third property, *plausibility*, is whether an explanation makes sense to the user reading it. This one cannot be computed, so it is treated here as something to design for. The thesis builds an interactive interface that shows all three methods side by side, derives its requirements from the human-computer interaction literature, and specifies the user study that would ask whether readers find the explanations useful, clear, and trustworthy.

The work is guided by two research questions:

RQ1 How do attention-, gradient-, and LOO-based explanations differ in faithfulness and robustness for LLM outputs?

RQ2 How should multi-method explanations be presented to a user, and how could that presentation be evaluated?

The contribution is therefore of two kinds. The first is a measured comparison of the three explanation families on a decoder-only model, on a common explanation target, reporting faithfulness and robustness together with the disagreement between the methods. The second is a working interface and a study design: an implementation that makes the three explanations and their differences visible to a reader, and a specification of the experiment that would test whether that helps.

The rest of the thesis is organised as follows. Chapter 2 introduces the two subjects the work rests on: how a transformer language model turns text into predictions, and what explainable AI takes an explanation to be. Chapter 3 fixes the explanation target for a generating model and presents the three families of method in turn, with the decisions each one requires and the limitations each one has. Chapter 4 describes the implementation, the faithfulness and robustness metrics, the prompts,

and the hypotheses, and reports what the measurements show. Chapter 5 derives the interface requirements from the human-computer interaction literature, shows the resulting system, and specifies the user study. Chapter 6 concludes the work by answering the two research questions and stating what the work does and does not establish.

2 Background: Explainable AI and Large Language Models

This chapter introduces the background required to follow the rest of this thesis. It covers two subjects. The first is the language model itself: how text is turned into numbers, and how the attention mechanism that the explanation methods of Chapter 3 read from is computed. The second is explainable AI, the field that studies what an explanation is and how one is judged.

2.1 Transformers and Autoregressive Language Models

A language model is a neural network which is trained to predict how a text continues. Given a text, it assigns a probability to every possible word that could come next, picks one, appends it, and repeats. Predicting a continuation is a task that needs no labelled data [24]. In any document, the continuation of each passage is simply the next word, so every document is already a collection of training examples. This is what allows such models to be trained on enormous amounts of training data without anyone labelling it, and a language model trained at that scale is what the term *large language model* (LLM) refers to.

That scale is only reachable if the text can be processed quickly, and this is what the architecture underneath these models provides. The *transformer*, introduced by Vaswani et al. [32] for machine translation, is a network of stacked identical layers, where each layer lets every word in the sentence exchange information with every other word before passing its result to the next layer. The recurrent networks it replaced read a text one word at a time, and the computation at each word needs the result of the one before it. A transformer computes all words at once [32], which is what makes training on very large collections of text practical.

A neural network works with numbers, not with text, so before any of this can happen the input has to be split into units from a fixed vocabulary and each unit turned into a number. Those units are *tokens*. A token can be a whole word, but also part of one, or a punctuation mark. Splitting the text is the job of a tokenizer, which

turns an input into a sequence of tokens $x = (x_1, \ldots, x_n)$, where each x_i is one entry of the model's vocabulary. A word like "unbelievable" can be split into "un", "believ" and "able", which is what lets a fixed vocabulary cover words it was never shown. How small those pieces are is a design choice. GPT-2, an autoregressive language model released by OpenAI in 2019 [24], builds its vocabulary up from single bytes, and since there are only 256 of them, it can represent any input, including characters the model has never seen.

Then, a token is converted into a numerical representation called an *embedding*. The embedding of token x_i is written $e_i \in \mathbb{R}^{d_{\text{model}}}$, where the values are learned during training. The number d_{model} determines the size of every token representation and is fixed when the model is defined. But an embedding does not carry position. It depends on the token alone, so the same word receives the same vector wherever it occurs in the sentence. Vaswani et al. [32] add a positional vector to each embedding, and what the model receives then is a sequence of vectors $(e_1, e_2, \ldots, e_n)$, where each vector is carrying both a token and its place in the input.

These vectors, however, still say nothing about the context around the token. Take the sentence "the trophy did not fit into the suitcase because it was too large." Here "it" is the trophy, since a trophy that is too large cannot fit. Change "large" to "small", and "it" becomes the suitcase. The embedding of "it" is the same vector in both sentences, since each embedding depends on its token alone, so nothing in e_i distinguishes the two readings. The model needs a mechanism that turns the context-free embedding e_i into a *representation* of the token that also reflects the tokens around it. Self-attention is exactly that mechanism. It builds the representation of a token as a weighted sum over all positions in the sequence, meaning the tokens $x_1, \ldots, x_n$ the model currently sees. The weight given to each position, its *attention weight*, states how much that position contributes to the new representation: tokens with higher weights contribute more, while tokens with weights close to zero contribute very little.

Each token representation is then transformed into three vectors called a *query*, *key*, and *value* using learned projection matrices [32]. Consider two positions i and j. The query of token x_i represents the information that token is looking for, and the key of token x_j represents the information that token offers. Their dot product is therefore a score for how well x_j matches what x_i needs, and the value of x_j is the information passed on when that score is high. The queries, keys and values of the whole sequence are then stacked into matrices Q, K and V, so that a matrix multiplication $QK^\top$ produces a score for every query-key pair. The scores are scaled down by $\sqrt{d_k}$, the dimension of the queries and keys, and the softmax then turns them into the attention weights. The updated representation of a position is the sum of the values, weighted by the attention weights of that position. All three matrices

are computed from the same sequence, so every position attends over the sequence it is itself part of, which is what makes this *self*-attention. Vaswani et al. [32] call it scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \tag{2.1}$$

As d_k increases, the dot products can also become large. Vaswani et al. [32] suspect that this pushes the softmax into regions where its gradients become very small, which makes learning harder. The fix is to divide the scores by $\sqrt{d_k}$ before the softmax is applied. That keeps them in a range where the gradients are still usable.

Because the weights are produced by a softmax, they are non-negative and sum to one for each position: what a token attends to is a distribution over the input. That is already the shape an importance score takes, which is why attention weights can be read as one at all.

A single round of attention gives each position one weighted average. So, every relationship in the sentence has to pass through the same set of weights, but the sentence holds several kinds of relationship at once - the subject a verb belongs to or the noun a pronoun refers to. And one average over all of them blurs each into the others. Transformers therefore run several attention operations in parallel, each called an attention *head*. Each head has its own learned projection matrices, so each derives its own queries, keys and values from the same input and can compare tokens. The outputs of these heads are concatenated and projected back to d_{model} [32]. Running them in parallel widens a single round, but it is still a single round. The tokens a position attends to are still context-free embeddings, so when “it” attends to “large”, what reaches it is “large” alone, not yet informed by “fit” or “suitcase”. Therefore, transformers also stack layers, each taking the representations produced by the layer below as its input and applying a feed-forward network to every position after the attention step. Later layers work on representations that already carry context, so that a position can be influenced by tokens it never attends to directly. A model is then described by the number of layers L it stacks and the number of heads H each layer holds. $\alpha_{ij}^{(l,h)}$ denotes the attention weight from position i to position j in head h of layer l.

But the transformer is an architecture, not a language model. What makes it one is the task this section began with - predicting how a text continues. A model doing that is called *autoregressive*: it produces its output one token at a time, predicting each from the tokens that came before it, so its own output becomes part of the input it reads. Self-attention as described so far cannot do this, because it lets every position attend to every other one in both directions. It has to be restricted, so that

the prediction for a position can depend only on the positions before it. Vaswani et al. [32] built the transformer from two stacks of such layers, following the encoder-decoder structure standard. The *encoder* reads the source sentence and builds a representation of it, with every position attending over the full sentence. The *decoder* stack writes the translation, and it is the autoregressive one. Its self-attention is *masked*: the scores for later positions are set to $-\infty$ before the softmax, which is how that restriction is implemented [32]. Two sequences are involved, then: one that is being read and one that is being produced.

A language model has no second sequence to encode. Its prompt might look like one, since it is given rather than generated, but it is not separate. The prompt is the beginning of the token sequence that the model then continues [24]. Self-attention makes no distinction between it and the tokens the model has produced itself, since both are positions in that one sequence. So, without a second sequence the encoder has nothing to do, and a model built just from the decoder stack is called *decoder-only*.

A model that generates one token at a time defines a probability over the whole sequence $x = (x_1, \ldots, x_n)$, prompt and generated tokens together. Language has a natural sequential order, so that probability is commonly factorised into the conditional probabilities of its tokens [24], and an autoregressive model computes one of these factors per step:

$$p(x) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1}). \tag{2.2}$$

This is what a decoder-only model produces, and it is where the problem this thesis works on begins. Every factor is a distribution over the whole vocabulary, one probability for every word the model knows, recomputed at every step. So there is no single number to point at. A classifier hands an explanation method an obvious target, its score for the predicted class, and a generative model has nothing of the kind. What the user reads is neither a distribution nor a product of factors. It is the answer, and the answer is what an explanation has to describe. Tracing it back to the words of the prompt is what lets a reader tell whether a fluent reply came from the input or from somewhere in the model itself.

2.2 Explainable Artificial Intelligence (XAI)

A model that cannot be explained can still be used, but its decisions cannot be checked. Nobody can ask why a loan was refused, whether a medical flag rested on the symptom or on the patient's postcode, or which part of a prompt produced an

answer that turned out to be wrong. Explaining what a model produces, and telling a good explanation from a misleading one, are the questions that gave rise to a field of its own, *explainable artificial intelligence*, which supplies the terms the rest of this thesis uses to classify attention weights, gradients and leave-one-out.

Explainable Artificial Intelligence (XAI) aims to make the decisions of models understandable to people. The need for it comes from deployment. A model whose reasoning stays hidden cannot be debugged when it fails, and cannot be trusted where a wrong answer does real harm. The need also appears in discussions of regulation. The European Union's General Data Protection Regulation, for example, is commonly interpreted as providing a "right to an explanation" for those affected by automated decisions [16]. Different user groups, however, want different things from an explanation [20]: a model developer looks for the bug, a regulatory body for the protected attribute, an affected user for a reason to accept or reject the answer. All of these uses rest on the same condition, that the explanation reflects what the model actually did. This condition is not met automatically. Judging a model means asking two separate questions: what is the answer based on, and how often that answer is right. The usual measure, *accuracy*, is the percentage of correct predictions, and it answers only the second. A model can score well on it while relying on something that has nothing to do with the task. Ribeiro et al. [25] compare two versions of one classifier, and the worse model was more accurate. The task was distinguishing posts about Christianity from posts about Atheism, and the first version reached 94 % on the benchmark. Its explanations gave weight to message-header words such as `Posting` or `Host`, which have nothing to do with either topic. On a religion dataset collected separately it fell to 57 %. Retrained on data with those artifacts removed, the same classifier scored lower on the benchmark, 88.6 %, and higher on the real data, 69 % [25]. So, accuracy alone ranked the two the wrong way, and what separated them was visible only in the explanations.

Not every model needs a separate explanation. An *interpretable* model is one whose prediction mechanism is obvious from its internal architecture or working principle [19]. A decision tree is a standard example, because its structure is itself the description of what it produces. Classifying an instance means following a path from the root down to a leaf where every step is a test on one feature, so reading the conditions along that path gives the complete reason for the output. This holds only while the tree stays small. What makes a decision tree interpretable is the size of the particular tree, not the fact that it is a decision tree [19]. A tree with thousands of nodes has paths hundreds of tests long, and reading one will probably not tell anyone anything.

But most models are not like this and *explainability* is what covers them. It is the ability of a model to give a description of what affected the decision [19], delivered

by a separate human-readable object such as a heatmap over the important words or an explanation in natural language. The two terms are often used as synonyms, but there is a difference between a model that can be read directly and a model that needs a separate explanation, and this thesis is concerned with the second case. In the three methods compared here, an explanation is a vector of scores, one number per word of the user's input, saying how much each word contributed to the answer the model produced. The three differ in how they arrive at those numbers, but the object they hand back is the same, which is what makes comparing them possible.

But an explanation can still be wrong about the model it describes. That is why it is possible to ask whether an explanation is *faithful*: whether it actually represents the reasoning behind the model's output [7]. Section 4.2 makes that precise, together with robustness and plausibility, the other two criteria this thesis judges explanations by.

2.3 Placing the Three Families

Surveys classify explanation methods along several axes [19, 16, 9]. Three of them are enough for the families compared here. Each one asks something different of a method: when it is applied, how much of the model it explains, and how much of the model it needs.

The first axis is **stage**: when the method is applied. Ante-hoc methods are used while the model is being trained and developed, in order to make it more transparent, whereas post-hoc methods are applied once the model has already been trained [16]. The model used in this thesis is pretrained, which makes every method post-hoc.

The second axis is **scope**. A local method explains the prediction made on one specific input, while a global method provides an overview or a complete description of the model as a whole [16]. This thesis explains individual answers to the user, and a description of the model as a whole does not answer why one particular answer was produced. Every explanation in this thesis is therefore local.

The third axis is **model specificity**. Model-specific approaches explain a particular model based on its internal structure and design, while model-agnostic approaches explain a model without depending on any understanding of that internal structure [16]. Gradient methods need the model to be differentiable and attention methods need a model that computes attention weights, so both are model-specific. Leave-one-out only replaces parts of the input and compares the outputs, so it is model-agnostic.

Access is easily confused with model specificity, but it classifies the situation rather than the method. A model is a white box if its parameters and architecture are known, and a black box if they are hidden from the user [9]. Attention and gradient methods read internal layers and parameters, so they need a white box model. Architecture and access can come apart in practice. A model served through an API is a black box to whoever is using it [9]: the architecture supports attention and gradient methods, but nothing outside the provider can reach the weights. A method is usable only where the deployment exposes what the method needs.

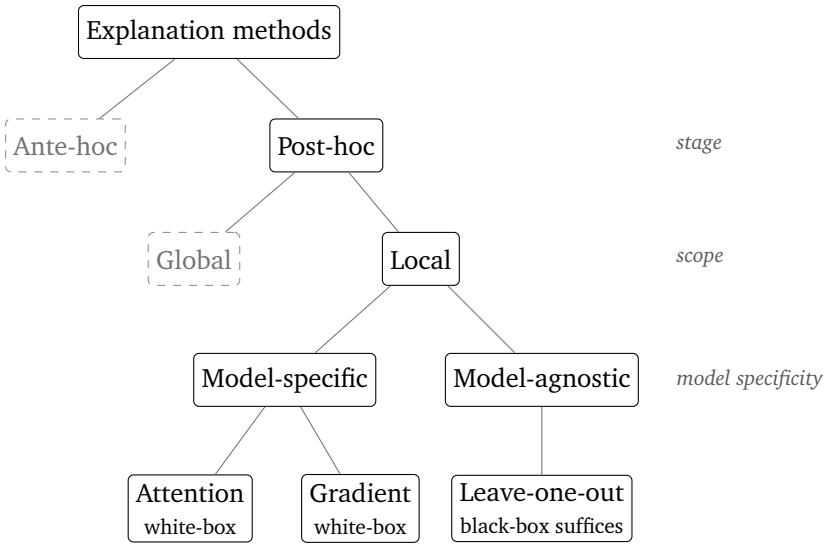


Fig. 2.1: The three axes along which explanation methods are classified, and the branch this thesis works in. Dashed boxes mark branches no method compared here occupies.

Figure 2.1 shows the axes as successive choices, and Table 2.1 shows where the three families sit.

Tab. 2.1: Placement of the three compared method families. The access column gives the least each family needs.

Family	Stage	Scope	Model specificity	Access
Attention	Post-hoc	Local	Model-specific (transformer)	White-box
Gradient	Post-hoc	Local	Model-specific (differentiable)	White-box
Leave-one-out	Post-hoc	Local	Model-agnostic	Black-box

Which explanation is right is not a question the taxonomy can answer. That depends on whether an explanation matches what the model actually did, and before it can be measured there is a prior question: what an explanation of a generating model attaches itself to.

3 Explanation Methods for Language Models

An explanation method scores the input against one output, and a text classifier supplies that output directly. It reads the input once and emits one score per class, so a single distribution of importance over the input explains it. A generating model does not provide such a score. At every step it produces a distribution over its whole vocabulary, the full set of tokens it can emit, and Equation 2.2 puts a single number on the finished sequence that only says how probable that text was. But what the user reads is neither of those. It is the answer as a whole, and that is what an explanation has to be about.

A decoder-only transformer of the kind described in Section 2.1 is large, distributed and sequential, and each of those three properties narrows the options. Scale puts the model itself out of reach. A model of this kind has billions of parameters spread over many layers, and nobody can read a description of what all of them do together. What can be explained instead is a single answer to a single input - a *local* explanation, in the sense Section 2.3 gave the term. Distribution puts a single located explanation out of reach. There is no component inside the model where the reasoning happens: every layer contributes and each one holds many heads, so a model of L layers with H heads per layer runs $L \times H$ attention operations in a single forward pass. Whatever is read out of the model therefore has to be aggregated over components. Sequential generation rules out explaining one step at a time. A score for a single generation step describes that step, not the reply the user reads.

This is the explanation target used throughout this thesis. An explanation assigns an importance score to each word the user supplied, for one answer the model generated from it: local, because it is about that one answer; aggregated, because the score has to be collected from across the model rather than read off one component; and about the answer as a whole rather than any single step.

Three families of method produce those scores, and this is where they begin to disagree. Two methods need a number that says how strongly the model supported the answer it gave, and they use the probability the model assigned to that answer. Gradients change a word's embedding slightly and measure how far that probability moves. Removal deletes the word from the prompt and measures how far the

probability falls without it. Attention needs no such number: its weights are read straight out of the model, which computes them on every forward pass as it decides how much each word should draw on the others.

3.1 Attention-based Explanations

Attention is the only one of the three that the model produces on its own. The weights are already there after a forward pass, already non-negative and summing to one over the input, which is why they have the shape of an importance score already. That made them the first candidate explanation to be examined closely.

Jain and Wallace [14] studied whether the attention weights of a trained model can be treated as an explanation of which input tokens mattered. Their work is also especially relevant here because it is the closest study to the comparison made in this thesis, and the only one reviewed for it that directly compares all three method families.

Their first experiment is on whether attention agrees with two other methods. Agreement is measured with Kendall's τ, a rank correlation. It compares the orderings two methods produce by going through every pair of words and asking whether both put that pair in the same order. For example, both rank *password* above *the*. The two methods agree about that pair, and it counts as concordant. If one ranks *password* higher and the other ranks *the* higher, they disagree, and the pair is discordant. Kendall's τ counts the concordant pairs, subtracts the discordant ones, and divides by the total number of pairs. Two orderings that agree on every pair give 1, and one that is the exact reverse of the other gives -1. When the two kinds of pair balance out, τ is 0, which is what two unrelated orderings look like. Only the ordering counts, which means that two methods that score a word 0.9 and 0.02 still agree perfectly, as long as both rank it first.

If attention is a trustworthy explanation, a token it weights highly should score highly under the other two methods as well. It does not. Against gradients it reaches τ between 0.08 and 0.47 across their datasets, and against leave-one-out between 0.05 and 0.44 - closer to two unrelated orderings than to two that match. Gradients and leave-one-out, on the other hand, agree with each other more than either agrees with attention: by more than 0.2 τ above attention's agreement with leave-one-out, and by about 0.25 above its agreement with gradients. That gap is the finding. The gradient-based method takes a derivative and leave-one-out deletes words and reruns the model, so there is no reason for them to land on the same ordering more often than attention does, and they do anyway. Attention, which is supposed to

be measuring the same thing, does not join them. The authors raise an objection to their own measure and answer it. Kendall's τ weighs every pair equally, and in a long review most words are irrelevant to the sentiment. Both methods score them near zero, so which one ranks higher is decided by differences too small to carry meaning, and those pairs pull τ towards 0. A low correlation might therefore say more about the many irrelevant words than about attention. The irrelevant words belong to the dataset and not to the model, so if that were the whole story, swapping the encoder would leave the correlations where they are. It does not. Their bidirectional LSTM is a recurrent encoder, reading the sentence one word at a time and carrying what it has seen forward, run once forwards and once backwards, so every word's representation is built from the whole sentence around it. Their other encoder does without that: it projects each word on its own, with no information from its neighbours. Attention over the second agrees with leave-one-out about 0.375 higher than attention over the first. The same metric and the same irrelevant words produce a different result, so it is the recurrent encoder's attention that the low numbers are about [14].

In the second experiment, instead of asking whether attention weights agree with other methods, they asked whether the attention distribution could have been different while the model's prediction is the same. First, they randomly rearranged the attention weights, and in most cases the model still produced a very similar prediction. Then they used an adversarial search to find attention distributions that were as different as possible from the original while still producing an equivalent prediction. Adversarial means the search looks for the worst case: not a distribution that is different, but the most different one the model will still accept. They were able to find such alternative attention distributions without much difficulty for most datasets. If the model can have a completely different attention map but still give the same answer, then the original attention map cannot confidently be considered the reason the model produced that answer [14].

Their conclusion is that attention weights do not provide faithful explanations and should not be presented as if they did. Two limits on that conclusion matter here: they used bidirectional LSTMs and not transformers, and every task was classification - so the model produces one score per class, and every explanation method has an obvious target to point at [14].

Wiegreffe and Pinter [38] responded directly, on the same bidirectional LSTM classifiers and the same datasets, and their disagreement is less about the measurements than about what was being asked of attention. Their idea is that attention is *an* explanation and not *the* explanation, meaning attention can provide one meaningful account of the prediction while other explanations can also be valid.

Two of their four experiments matter the most. The first is a diagnostic. They strip a classifier of everything that reads context: the LSTM is replaced by a layer that processes each token on its own, so the model cannot tell from the sentence which words matter. The only thing left that could tell it is the attention distribution, which is supplied from outside as a fixed set of weights and used to combine the per-token outputs, during training and testing alike. If a set of weights makes this model classify better, the information has to have come from the weights, because there is nowhere else it could come from. Four sets of weights are put through this test, and they rank consistently: weights taken from the trained LSTM beat weights the stripped model is allowed to learn for itself, which in turn beat a uniform baseline, on all four datasets - though on three of them the first margin is under 0.02. Adversarially obtained distributions come last of the informative guides on three datasets. On Anemia, a medical dataset of hospital discharge summaries, they match the trained weights, which the authors put down to a heavily skewed label distribution. An attention distribution therefore carries something about token importance that is not an artefact of the model that produced it, since it remains useful to a model with no access to context at all [38].

The second experiment is about how those adversarial distributions were found. Wiegreffe and Pinter [38] trained a whole adversarial model, rewarding it for predicting like the original while attending differently, where Jain and Wallace [14] searched for a different distribution per example with the model held fixed. One set of parameters then has to serve every instance, which is far less freedom than choosing a distribution per example. Adversaries of this kind can still be found, but how far they get varies by dataset, their distributions are less extreme, and they perform poorly in the diagnostic above. An attention map that can be manipulated one example at a time is therefore weaker evidence than one that could be replaced systematically [38].

Neither side wins outright. Wiegreffe and Pinter [38] accept the correlation experiment and dispute only what the adversarial one establishes: an attention distribution carries information relevant to the prediction, but carrying relevant information is not the same as identifying the tokens that caused it, and alternative distributions do still exist. What the exchange settles is narrower than either title suggests. This thesis does not need the verdict either way - it needs to know how attention's scores compare with two methods that measure something else entirely.

Both papers argued over recurrent classifiers, where one attention layer produces one distribution over the input. Reading that as importance needs no further work, since the distribution already holds one number per token. A decoder-only model produces far more than one. Every layer holds several heads and each head computes its own attention, so a model of L layers with H heads already yields $L \times H$ distributions per

position. On top of that, the model attends once for every token it generates, and each of those attends over the whole sequence: the weight $\alpha_{ij}^{(l,h)}$ of Section 2.1 exists for every pair of positions i and j, in every head, in every layer. What a forward pass leaves behind is therefore an $L \times H \times n \times n$ tensor for a sequence of n tokens. Producing one score per prompt word from that tensor means deciding which of those weights to read, and how to reduce what is read to one number per word. Those decisions are what this section is about.

Which of those weights matter follows from the explanation target. Collecting the weights of one head into a grid, one row per position, gives that head's *attention matrix*: row i holds the distribution of position i over the sequence, so the rows stand for the positions doing the attending and the columns for the positions attended to. The explanation scores prompt words for one generated answer, so of that grid only the rows belonging to generated tokens and the columns belonging to prompt words are of interest. The result is a block of rows, one per generated token, each holding that token's attention over the prompt. Reading only the last row is a cheap route and it was used in an earlier version of this work, but it makes the explanation depend on whichever token happens to end the answer, which is typically an end-of-sequence marker. It also puts attention on a different target from the other two methods, since both score the whole answer. So part of the disagreement between them could follow from that choice. The scores used here are pooled over all generated positions instead, with special tokens excluded, and these scores are what the evaluation measures. The same weights can also be shown without the pooling step, one row per word of the answer, so that a reader can watch the answer's attention shift from one output word to the next.

Which layer is being read is not a free parameter either, because attention does not carry the same information at every layer. Abnar and Zuidema [1] find that the attention patterns are different between positions in the first few layers, and that higher up the weights are close to uniform across the input. In those early layers tokens are still relatively distinct from each other, so a content word can attract visibly more attention than a function word - "password" over "the" or "is", for example. Deeper in the model each token's representation already contains information drawn from many other tokens, which is why the weights flatten: the embeddings have stopped being identifiable with the tokens they started as [1]. This does not mean that there are no important words, only that the difference between them is no longer visible in the weights.

Abnar and Zuidema [1] answer this by combining the layers. The first method, attention rollout, multiplies the attention matrices of successive layers together. The second one, attention flow, treats the model as a graph whose nodes are the input tokens and the hidden embeddings above them, with attention weights as edges, then

uses a maximum-flow computation to measure how much of the information in a hidden embedding higher up could have come from a given input token [1]. Neither method can ignore that attention is not the only route through the model: each layer also passes a token's representation forward directly and adds the attention output on top of it, the path known as the residual stream, so both methods add an identity term at every layer to keep that path in the calculation. Abnar and Zuidema [1] find that the scores agree better with input gradients and with blank-out than reading a single layer does. Blank-out is a variant of leave-one-out that replaces the word with `UNK`, the symbol a model uses for words outside its vocabulary, instead of removing it from the sequence. That result rests on one task and one model, with a smaller check on a second model scored against gradients alone, so whether rollout and flow are generally better is not clear.

This thesis reads three separate layers: an early, a middle and a late one, at indices $\min(2, n-1)$, $\lfloor n/2 \rfloor$ and $n-1$ for a model of n layers. Their scores are shown side by side. The reason is that the flattening described above is itself something a user should be able to see, and rollout and flow, by design, return a single score in which it no longer appears. The choice is diagnostic: it keeps the comparison between depths visible instead of resolving it.

The heads inside a layer are not kept apart in the same way. Every head applies its own query projection, so each compares tokens on its own terms [32], and their attention matrices differ accordingly. The scores used in this thesis are a plain unweighted mean over all heads of the chosen layer. Alternatives exist - taking the largest weight across heads, reading one selected head or weighting heads by some measure of importance - but each of them asserts that some heads matter more than others, and nothing in this setup says which.

Averaging over heads leaves one score per token, but the explanation is presented over words, and the tokenizer does not guarantee that these are the same thing. A word may be split into several subtokens, so the scores of a word's subtokens have to be reduced to one number. The approach used in this thesis is the maximum over them: attention weights are non-negative, so it keeps the strongest attention the word received anywhere in its parts. Summing instead would make a word's score grow with how many pieces the tokenizer happened to produce, averaging would pull a word down for having one weakly attended piece, and taking the first or the last subtoken discards the rest on no principled ground.

None of those decisions changes what attention measures. An attention weight says how much one position contributed to another's representation as the model built it. That is a fact about the model's internal routing, not about the answer: a word can be attended to heavily and still make no difference to what the model said, and a word

attended to weakly can be the one the answer turns on. Aggregating over heads or layers does not repair this, because no amount of combining changes what the numbers were measuring. There is still a difference between the two aggregations, and it is worth stating. The depth comparison survives into the measurements of Chapter 4, which score the early, middle and late layers separately. The head axis does not: the mean is taken once and individual heads are never scored. Since each head compares tokens on its own terms, a head that tracked the words the answer turns on is averaged together with every head that did not, and what is reported is that average. Singling one out would require a criterion for which head matters, and this setup has none - but nothing measured here shows that the average head stands in for any particular one.

3.2 Gradient-based Explanations

Gradients ask a different question: if this input word was changed slightly, how much would the answer change?

Li et al. [15] brought first-derivative saliency to natural language processing. Their definition assumes a model that scores an input against one target label c - a sentiment label, a part-of-speech tag, or the index of the next word to predict - and takes the saliency of an input embedding e to be the magnitude of the derivative of that score S_c:

$$w(e) = \left.\frac{\partial S_c}{\partial e}\right|_e, \qquad \text{saliency}(e) = |w(e)|. \tag{3.1}$$

A large value means a small change to the word could move the score considerably, which is read as the word being important. A small value is read as the word mattering little.

A decoder-only model producing a whole answer has one target per generated position, so there is no single S_c to differentiate and one has to be chosen.

That anchoring is what makes gradients worth comparing against attention. Attention weights are quantities produced inside the model and they describe how information is distributed between tokens during the forward pass. A gradient is defined against the model's output, so it answers a question about the answer and not about the routing that produced it. It costs one backward pass per explanation, where reading attention costs nothing beyond the forward pass that has already happened.

A gradient measures the model's local sensitivity at the current input only, and that is a problem because neural networks become saturated. That means that once the input into a ReLU or sigmoid function grows past some point, there will be no change in the output. Stack enough layers of these functions and a confident prediction ends up in a region where the gradient is near zero. Therefore, a word the answer depends on can score almost nothing just because the model is already sure. Sundararajan et al. [30] make the point with $f(x) = 1 - \mathrm{ReLU}(1 - x)$: between the baseline $x = 0$ and the input $x = 2$ the output rises from 0 to 1, so the feature matters, but the function has become flat at $x = 1$ and the gradient at the input is exactly 0. This breaks the *Sensitivity* axiom, which says that when an input and a baseline differ in one feature and the model predicts differently for them, that feature should be given a non-zero score. This gap is also visible in text. Li et al. [15] tested the sentence *I hate the movie though the plot is interesting*. Their recurrent model scored the second clause highly and gave *I hate the movie* almost nothing - the saliency map, the per-word scores of Equation 3.1 laid over the sentence, was near zero exactly where the sentiment was. The model was, however, using that first clause: it classified the sentence as very negative, and when the second clause was tested on its own - the same model labelled it positive. That map was therefore silent about the words the answer depended on.

The two failures have different causes: the ReLU network is flat at the input, while the recurrent model is not flat at all and simply cannot show how two clauses combine in a first-order derivative. Both follow from a derivative describing the model only at the point where it is taken.

Escaping that single point is what most of the work on gradient-based explanations after Li et al. [15] is about, and the escapes are worth separating before choosing one. Wang et al. [36] review the gradient family and sort it into four classes - vanilla gradients, integrated gradients, bias gradients and post-processing.

Vanilla gradients evaluate the derivative at the input and use it directly. Two variants within the class modify that derivative instead of replacing it. Gradient $\times$ Input multiplies it by the input value, and Guided Backpropagation zeroes every entry that is negative in both forward and backward passes [36]. That last filter costs information, and Wang et al. [36] state that a method that discards negative signal cannot identify a feature's negative influence on the output, so a word arguing against the model's answer looks no different from an irrelevant one. The class is the cheapest of the four and needs just one backward pass, but saturation defeats all of it.

Integrated gradients exists specifically to fix saturation [36], and this is the class the thesis draws from. Sundararajan et al. [30] introduced Integrated Gradients by

stating two properties any attribution method should have, then building a method that satisfies both. Both are defined against a *baseline*: an input that stands for the absence of what is being scored, chosen so the model's prediction on it is neutral. Their example is the black image for an object recogniser. *Sensitivity* then says that if an input and the baseline differ in one feature and the model predicts differently for them, that feature must be given a non-zero score. *Implementation Invariance* says that two networks computing the same function must receive the same attributions, however differently they are built, since what a word contributed to an answer cannot depend on how the network was coded.

No method before theirs satisfied both. Ordinary gradients fail Sensitivity for the reason already seen: where the function has flattened, the gradient is zero even though the feature mattered. But they do satisfy Implementation Invariance, and without any effort. A gradient can be computed straight from output to input, or by the chain rule through whatever the network does in between, and the two give the same answer [30]. How the network is built therefore cannot affect the attribution.

Two methods that work by passing relevance back through the network layer by layer, Layer-wise Relevance Propagation and DeepLift, fix Sensitivity but lose the other axiom. Instead of taking the gradient at the input, they compare the input against the baseline directly, a substitution Sundararajan et al. [30] call a discrete gradient. The chain rule does not hold for discrete gradients, so the guarantee disappears: two networks computing the same function can be given different attributions.

Integrated Gradients keeps both by integrating the gradients along a straight-line path from the baseline to the actual input. Saturation stops being fatal, because the path runs through the region where the function was still responding even if it has gone flat by the time it reaches the input. What the accumulated gradients sum to is exactly $F(x) - F(x')$, the gap between the model's output at the input and at the baseline - the property Sundararajan et al. [30] call *completeness*. Sensitivity follows from it: a feature that alone accounts for a changed prediction receives that entire gap, so it cannot score zero. Implementation Invariance survives because the gradients being accumulated are ordinary ones. The method also leaves the network alone - no layer-by-layer rules to implement, just the standard gradient operation called at a series of points along the path.

A later result shows what those axioms are worth in practice. Wang et al. [35] merged a trained model with a network that overwhelmed its gradients while leaving its predictions untouched, and the saliency maps of the merged model then highlighted stop words or whichever token the attack chose. Integrated Gradients was the most resilient of the methods they tested, and the axiom is why: two models

with equal predictions on all inputs should have equal attributions. The merged model is only similar to the original rather than equivalent to it, so the protection is partial - with the original model's gradients regularised, Integrated Gradients is manipulated too. What matters here is the shape of the failure rather than the attack. An explanation changed while the model's behaviour did not, and a method that reports differently about behaviour that has not changed is unreliable whatever its faithfulness score says. That is a question about *robustness*, the second criterion this thesis measures, and it is the reason faithfulness alone is not enough to judge these methods by.

Instead of evaluating the gradient at a single point, Integrated Gradients accumulates gradients along a path from a neutral baseline x' to the actual input x [30]:

$$\mathrm{IG}_i(x) \;=\; (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F\big(x' + \alpha(x - x')\big)}{\partial x_i}\, d\alpha, \tag{3.2}$$

where F is the model, x' is a neutral input that carries no information, and $x_i - x_i'$ is how much feature i changed from the baseline. The variable α moves from 0 to 1 along the path, so $x' + \alpha(x - x')$ is a point between the baseline and the real input. The important detail is that the gradient is taken at these in-between points and not at x itself. The integral adds up all the gradients into one importance score, and it is approximated by a Riemann sum. This is how the saturation problem is solved: even if the function is already flat at x, the path still passes through regions where it is not, so a feature whose gradient is zero at the input still collects attribution along the way. Integrated Gradients therefore satisfy both *Sensitivity* and *Implementation Invariance*.

Bias gradients attribute the network's bias terms alongside input gradients, and the class handles the saturation as well. Where the gradient is small, the bias term contributes relatively more, so the biases matter most exactly where the gradients say least [36]. This class is not used in this thesis, since it is demonstrated on convolutional image models, and the transfer to text remains unknown [36].

Post-processing methods do not compute importance from scratch. Instead, they take an explanation from another method and either reduce its noise, as *SmoothGrad* does by averaging over noisy copies of the input, or measure how far it varies, as *VarGrad* does [36]. Wang et al. [36] treat the class as orthogonal to the other three: it wraps a method rather than replacing one, and its quality is inherited from whatever it wraps.

That leaves one more candidate, which the taxonomy does not cover: *Conservative LRP*, which is built specifically for transformers [4]. Layer-wise Relevance Propagation (LRP) treats the model's score for the answer as a fixed quantity and hands it

back through the network. Starting at the last layer, that score is split among the units that fed it, those units divide their part among the units below them, and so on until every input word has a share. The conservation axiom is what keeps the accounting straight: a layer hands down exactly what it was given, so the shares arriving at the input still sum up to the score at the output. Ali et al. [4] show that standard Gradient $\times$ Input breaks conservation at two components of a transformer, attention heads and LayerNorm, and repair it by changing how relevance passes through those two. In their evaluation it reached the best score on every dataset in their main tables, on both perturbation metrics they report. That makes it the strongest competitor to Integrated Gradients here, but it is not the method this thesis uses. The reason is that its propagation rules have to be written for each component they pass through. Ali et al. [4] write them for LayerNorm, which the model used here does not have. Llama, the model used in this thesis, normalises with RMSNorm instead. Both normalise a layer's output so its values stay in a usable range, but they compute the normalisation differently, so using the method here would mean assuming its rules carry over to a component they were not derived for. The same property carries a second cost. The propagation rules work by changing how the model computes its forward pass, so they have to be written for one particular implementation, and a tool meant to be pointed at more than one model would need that work again for every model it supports. Integrated Gradients asks only for ordinary gradients, so neither cost applies to it.

Table 3.1 sets out what each method does and what it costs.

The implementation described in Section 4.1 uses Integrated Gradients. The integral of Equation 3.2 cannot be computed exactly, so it is approximated numerically. The number of steps is settled there.

Equation 3.1 needs a scalar S_c and a decoder-only model does not supply one, so the choice made here is the summed log-probability of the whole generated response. All three methods are defined on that same response. Attention scores are pooled over the positions that produce it, and Integrated Gradients and leave-one-out both reduce it to a log-probability. But these two methods reduce the response differently. Integrated Gradients sums the per-token values and leave-one-out averages them. For a recorded answer of a fixed length the sum is the mean times the number of tokens, and scaling every word by the same factor leaves their order unchanged, so the difference reaches none of the comparisons in Section 4.7.

A shared target is still not enough, because the three methods do not return their scores on the same units. Integrated Gradients produces one value per embedding dimension, which is summed to a value per subword, and each word then takes its largest subword by absolute value. Leave-one-out needs neither step, since it

Tab. 3.1: Gradient-based attribution methods, ordered by the four classes of Wang et al. [36]. Post-processing is listed after the other classes because it wraps them rather than competing with them. Conservative LRP is listed last because it does not belong to this taxonomy.

Class	Method	Idea and principal trade-off	Ref.
Vanilla gradients	Plain gradient (saliency), Gradient $\times$ Input	Evaluate the derivative at the input. Cheapest of all; violates Sensitivity under saturation, and Gradient $\times$ Input additionally breaks conservation inside Transformers	[15, 4]
Integrated gradients	Integrated Gradients, Expected Gradients	Accumulate gradients along a path from a baseline to the input. Strongest axiomatic guarantees; cost scales with the number of integration steps, and the result depends on the baseline	[30, 36]
Bias gradients	FullGrad	Attribute the bias terms as well as the input gradients. Covers what gradients miss in deep layers; demonstrated on convolutional image models, and the transfer to text is stated to be unknown	[36]
Post-processing	SmoothGrad, VarGrad	Average or take the variance of an attribution over noised inputs. Reduces visual noise; does not repair the underlying signal, and its quality is inherited from the method it wraps	[36]
(outside the taxonomy)	Conservative LRP	Restore conservation at attention heads and LayerNorm by locally linearising both. Best reported empirical scores at the cost of a single backward pass; requires modifying the model's forward computation	[4]

deletes a whole word's subtokens at once and scores the word directly, and attention is pooled over the positions of the answer. The scores of the two signed methods are divided by their largest absolute value, which puts the most decisive word at ± 1 and keeps the direction: a negative score marks a word that pushed the model away from the answer it gave.

Two limitations come with the choice of Integrated Gradients.

The first is the zero baseline. Integrated Gradients needs a reference input, and a zero embedding is used here. Sundararajan et al. [30] recommend exactly this for text. Their reason is that training gives unimportant words small norms, so in the limit an all-zero vector is what unimportance looks like. They admit it is not a valid input, since no token actually has that embedding, and defend it on the

mathematical grounds above. Whether that claim about embedding norms holds for the model used here is measured in Section 4.7.

The more serious limitation belongs to the family rather than to this setup. Wang et al. [35] showed that a model's gradients can be redirected to mislead. They take a trained model and merge it with a second network whose gradients are strong and point the wrong way, but whose predictions are too weak to change the answer. The merged model replies exactly as the original did. Its saliency map points wherever the attacker wanted. What makes this serious is who the attacker is: not an outsider feeding the model bad inputs, but the organisation that owns it, deceiving whoever audits it. Integrated Gradients holds up best of the methods they test. Implementation Invariance is the reason, since two models that behave alike should get the same attributions, and the merged model behaves almost exactly like the original. But that protection only goes so far, and a regularised version of the same attack gets past Integrated Gradients too [35].

3.3 Leave-One-Out Explanations

Integrated Gradients changes a word's embedding by a small amount and watches the score respond. The third family changes the input too, but it removes a word instead and compares the model's score with and without it.

Fong and Vedaldi [10] give a formulation of perturbation-based explanation: to find out what a model considered important, change parts of the input and watch what the output does. Leave-one-out is the text-domain case of that idea, deleting one word at a time and rerunning the model. What counts as deleting something is already a problem in their setting. They compare replacing an image region with a constant colour, with noise, or with a blurred copy of itself. None of the three shows what the image would look like without that region, and they have no way to produce one that does.

Deleting a word from text needs no such substitute, because the word can simply be dropped. What remains to be decided is what to measure, and here it is the model's score for a fixed answer. The model first generates once on the complete prompt, and the produced answer is then held fixed. For each word, that word is removed and the model is run again - not to generate another answer, but to evaluate the same original answer under the shortened prompt. This gives a probability for each token of the fixed answer, and those probabilities are combined into a single score for the response: the mean of their log-probabilities, averaged, not summed, so that the score does not grow with the length of the answer. If removing a word lowers

that score, the word had a strong effect on the model's support for the answer. If the score changes little, its removal had small effect. Allowing the model to generate a new answer after every deletion would instead require deciding how to measure the difference between two pieces of generated text. Holding the answer fixed avoids that problem and leaves one scalar quantity to compare. The disadvantage is the computational cost. The procedure removes one word at a time, so a prompt of n words requires n additional passes and the cost of an explanation grows linearly with the length of the input.

The described procedure is implemented directly, with one forward pass per word and no regeneration. Writing y for the fixed answer, the score of word i is

$$s_i \;=\; \log p(y \mid x) \;-\; \log p(y \mid x_{\setminus i}),$$

where x is the complete prompt, $x_{\setminus i}$ is the same prompt without the word i , and both log-probabilities are means over the tokens of y. The sign says which way the word pushed. A positive s_i means the answer got harder to produce once the word was gone, so the word had been supporting it. A negative one means the answer got easier, so the word had been working against it. A score near zero means the word made no difference in either direction.

The scores are divided by the largest absolute score in the prompt, which puts them on $[-1, 1]$ with the sign kept. This normalisation is used for display. The raw scores are differences in log-probability, and their absolute magnitude depends on the model, prompt, and generated response, so values from different prompts are not directly comparable.

Deleting the word is only one way to remove it, and different operators produce different scores [13, 10]. Replacing the word with a mask token is the standard alternative, but it assumes that the model was trained with such a mask and knows what to do with it. Padding has the same defect, and the model has to process a token that has no meaning in a position where a real word once stood. Noise substitutes an arbitrary vector for a real word, which measures the effect of corrupting the input rather than of losing the word. Replacing the word with a plausible alternative avoids all three problems, but it requires a second model to propose the replacement, which means that the explanation would depend on that model's choices as much as on the one that is being explained. Deletion is used here because it is the only one of the five that adds nothing to the input, so whatever the score measures comes from the word's absence rather than from what was put in its place.

Deletion does not introduce a replacement, but it still changes the structure of the input. Since the word's subtokens are spliced out of the sequence, what the model

receives is a shorter prompt, and it is not always well-formed. One prompt in the evaluation set is *The capital of France, not Berlin, is what city?*. Removing *Berlin* leaves *The capital of France, not, is what city?*, and this is not a sentence anyone would write. The model answers it anyway, and it answers with something that looks like an ordinary reply. And that is what makes the problem hard to see. The log-probability of the original answer falls, and there are two reasons it might have fallen: either the word was carrying information the answer needed, or the prompt stopped resembling the text the model was trained on. Leave-one-out sees a single number and cannot separate them, and the output gives no indication which case it is, so there is no signal to filter on. Deleting *not* from the same prompt is different. It leaves *The capital of France, Berlin, is what city?*. The sentence is perfectly well-formed but has a different meaning, so a large drop means the model was relying on that word. Deleting *Berlin* leaves a broken sentence, and a large drop there could mean the same thing or could mean only that the model was given something it was never trained on.

This problem has been discovered directly in image classification: an MNIST classifier assigns 91% confidence to an input of pure Gaussian noise, a result reported by Qiu et al. [23] from earlier work and taken there as proof that confidence says nothing about whether the input was one the model could meaningfully judge. Fong and Vedaldi [10] met the same behaviour from the other direction, searching for the image region that reduced a classifier's score the most: their optimisation kept finding unnatural patterns that fooled the model instead of regions that mattered to it, and they had to constrain the mask to keep it producing realistic edits. Both results come from images, and this thesis makes no equivalent measurement on text, so how far a deleted word pushes a prompt off-distribution is not something the numbers in Section 4.7 can answer.

That ambiguity is the out-of-distribution problem [23] - the shortened prompt is not like the text the model was trained on, so a change in output does not necessarily mean the removed word mattered. This problem has three known responses in the literature, and they intervene at different points. Fong and Vedaldi [10] constrain the explanation method so that it cannot produce strange patterns that can deceive the model. Qiu et al. [23], instead of restricting the modification, check whether the input is unnatural and, in that case, down-weight it. Zintgraf et al. [40], on the other hand, look for a replacement for the removed word based on the context of the sentence. This work uses none of these solutions. The word is just deleted and deletion is the most exposed of the available choices in that it makes no attempt to keep the shortened prompt close to the training distribution. So, whatever effect the problem has on the scores - it is present in full. No comparison against the alternatives was run, so nothing here shows that deletion is the better choice, it is a limitation of this work. But there is some reason to think the comparison would

not have settled the question. Šimić et al. [29] ran roughly $27{,}600$ perturbation experiments and found the best operator is changing with the dataset, the model architecture, and even the individual class being explained. Their setting is time-series classification, so their specific operators do not carry over to deleting a word from a prompt. But if no operator wins across that many runs, the choice is unlikely to have a general answer, and the most that can be asked of it is that it is stated and justified.

The linear cost is the first limitation, even though it does not bite here, because the prompts used in this thesis are short single questions. Attention and Integrated Gradients need a fixed number of model passes no matter how long the input is, while leave-one-out needs one extra pass for every word in the prompt. Longer inputs therefore cost all three methods more, and leave-one-out most of all.

Second, the operation that defines the method is also the operation used to measure faithfulness. The removal-based metrics used in Section 4.4 delete the tokens an explanation ranks highest and observe how far the output falls. That is what leave-one-out does to produce its scores in the first place: it deletes a word and records how far the output falls. The metric and one of the methods it grades are therefore built on the same operation, and leave-one-out should be expected to lead on it. The advantage is, however, bounded, because leave-one-out removes words one at a time while the metric removes the top-ranked set jointly, and the two come apart whenever words interact. But a strong result for leave-one-out here means less than the same result for attention or Integrated Gradients, neither of which is built from removals. The same is true of the out-of-distribution problem: the metric deletes tokens just as the method does, so both feed the model inputs unlike its training data, and neither can separate a word that mattered from a prompt that became strange.

3.4 Comparing the Three Families

Even though all three explanation methods assign an importance score to every input token, they may be measuring very different quantities. And although their outputs look similar, the way they define “importance” will differ and they will capture different aspects of how the model works. So, disagreement does not necessarily mean that one of them is wrong. Each of the three simply answers a different question about how the model produced its answer.

Attention weights are produced by the model during its forward pass, so reading them off costs nothing beyond running the model. They describe how strongly one

token attends to others within a particular layer and attention head (Section 2.1). In this sense, attention reflects the internal routing of information through the model. However, a token may receive a high attention weight without having a strong influence on the model's answer [14]. A gradient is taken with respect to the model's output, which ties it more closely to the answer than an attention weight, but it is a local quantity [30]. Equation 3.2 integrates the derivative of the model's score along a path from a baseline to the real input, so it indicates how the output responds to changes in each embedding. But the word itself never leaves the sequence. The model is never run on a sentence without it, so the score answers a question about small changes around the input, not about the input being different. Leave-one-out, however, removes one word from the input, runs the model again, and measures the difference in the log-probability of the original answer [10]. But, this creates a different sentence, so the reason for changes in the model's output may have nothing to do with the word's importance (Section 3.3).

So the three do not answer the same question. Attention measures internal allocation, gradients measure local sensitivity, and leave-one-out measures the effect of an intervention. Table 3.2 summarises the distinction.

Tab. 3.2: What each method family measures, what it needs from the model, and what it costs for an input of n words. Cost is counted in model passes, where a backward pass is roughly the cost of a forward one. m is the number of integration steps, fixed at 600 in Section 4.1.

Family	Quantity measured	Model access	Cost
Attention	Internal allocation of reads across positions	Attention weights	None beyond the forward pass
Gradient	Local sensitivity of the output to each embedding	Weights and gradients	m forward and backward passes
Leave-one-out	Change in the output when the word is actually removed	Output scores only	$n + 1$ forward passes

The model access column of Table 3.2 shows where each method can be applied at all. Attention and gradient methods need the model to be open, and gradients need it to be differentiable and to expose its backward pass. Leave-one-out needs neither though. It only needs the ability to provide an input and observe the output [25], which makes it suitable for models accessed through an external interface. This is the setting for many of the largest LLMs, and it is the practical reason for keeping the most expensive of the three families in the comparison (Section 2.2).

All three families have shown limitations, and they fail in different ways. Attention's problem is that the weights are not always the reason for the answer, since a different distribution over the same input can produce the same prediction [14]. The weights still tell us something about the model [38], but not reliably why it answered as it

did. The gradient scores can be made to point wherever a model's owner wants, while the predictions stay fixed [35]. Leave-one-out works as intended, but the input it constructs is one the model was never trained on [23]. One failure is about what the quantity means, one about who controls it, and one about what the measurement does to the input.

None of the three explanation methods can be treated as ground truth for evaluating the others. Jain and Wallace [14] read low agreement between attention and the other two as an argument against attention. That reading assumes the three answer the same question, so that a disagreement has to mean one of them is wrong. They do not answer the same question, and two measurements of different things can disagree without either being false. Each method also has a failure mode of its own, which is the second reason none of them can serve as the reference.

No study reviewed here compares the three families as a set on a generated answer. The metric work has the same limit. Chan et al. [8] test the faithfulness measures against each other and find comprehensiveness and sufficiency the strongest, but they do it on classifiers. Chapter 4 runs the comparison on a decoder-only model, choosing the criteria and the metrics to score it with as part of the work.

4 Computational Evaluation

No study has compared all three families against each other on a decoder-only model. Running the three methods on one would be the obvious way to start, but it does not close that gap: it produces three explanations of one answer with no way to say which one of them is better. A criterion is needed, and prompts on which that criterion can separate the methods - one can look faithful on a question with one obvious answer-carrying word and fail on a question that has none.

4.1 Implementation

Every number in this chapter comes from a single program, written in Python with PyTorch 2.11 [21] and Hugging Face Transformers 5.4 [39]. The runs reported below are driven by a command-line harness: a second program that reads the prompts from a file, calls the three methods on each of them, and writes what comes back to disk. It does not carry its own copy of the methods but imports them from the module the server also uses, so the code measured here is also the code that is reached by the user's question.

The model used in this thesis is Llama-3.2-1B-Instruct [17], a decoder-only autoregressive language model. One of the reasons why this model was chosen is that the weights have to be openly available, since two methods cannot run against an API: Attention scores have to be read out of the forward pass and Integrated Gradients differentiates through the embedding layer. Another reason is that the model also has to be small enough, so that six hundred forward and backward passes per prompt remain affordable, which a one-billion-parameter model on a single GPU manages and a larger one does not. This model is also an instruction-tuned version. That changes what the methods are given to work with, since a prompt does not reach the model as a bare text. It is instead wrapped in the chat template it was tuned on, with a fixed system message before the user's question and a marker after it where the reply begins. The system message itself asks for an answer of two to four sentences, so what it buys here is that the answer begins with content directly and not with a preamble.

This model is also different from the transformer described in Section 2.1 in how its attention heads are built, but that difference does not change the shape of the attention weights, which remain nonnegative, normalised over positions, and indexed by layer and attention head. In the original architecture each head derives its own queries, keys and values from the input. Here the 32 heads keep their own queries but share 8 sets of keys and values, four heads to a set, an arrangement known as grouped-query attention that is fixed in the published weights. The reason is the cost of generating text: the model reloads every key and value at each step it generates, and that memory traffic is what limits how fast it runs [3]. Sharing one set of keys and values across four heads cuts what has to be reloaded to a quarter, and Ainslie et al. [3] propose the arrangement as a middle point between a separate set per head and a single set for the whole layer, keeping most of the quality of the first at close to the speed of the second. Heads in a group therefore differ in what they ask for but not in what they compare against, which leaves them less independent than the original architecture assumes, and Section 3.1 averages across all 32.

Each of the three methods is a function that takes a prompt and returns one score per word. Chapter 3 defines what each of them computes. What follows now is what the code does with those definitions.

Attention takes two passes. The needed weights are the ones the model computes over the finished prompt and answer, and those do not exist when the answer is still being written: generation produces one token at a time, each pass seeing only what came before it. The answer is therefore generated first, and the prompt and answer are then fed back through the model as one sequence with attention outputs switched on. Nothing is generated on this second pass. Since the model's self-attention is masked, each position attends only to positions before it, so re-reading a finished sequence reproduces the weights of generation, and does not approximate them. Asking the model for its attention weights has a cost. For speed it normally computes the softmax without ever holding the full table of weights in memory, and requesting that table forces a slower implementation that builds it. What that implementation returns is one $n \times n$ table per head and layer, with n the combined length of prompt and answer, from which Section 3.1 takes the answer's rows against the prompt's columns, averages the heads, and keeps the early, middle and late layers apart.

Integrated Gradients needs three settings the method itself does not supply. The method fades the input in from a baseline to the real prompt and accumulates the gradient along the way, and Equation 3.2 writes that accumulation as an integral, which is approximated at the midpoints of 600 equal intervals along that path, and those 600 inputs go through the model in batches of eight, because six hundred copies of the sequence do not fit in memory at once. Six hundred is not a conventional

figure, and it was fixed by measurement. Completeness gives a check that can be run on any input: the scores should sum to $F(x) - F(x')$, so if the parts do not add up to the whole, the approximation is off and the gap measures by how much. The implementation therefore computes both sides on every call and records their ratio, which completeness puts at 1.0. That check was run over five prompts at four step counts. The best was 600, with a mean deviation $|\text{ratio} - 1|$ of 0.09, against 0.27 at 300 steps, 0.14 at 900 and 0.10 at 1200. Adding steps past 600 did not help. The second setting is the baseline the path starts from, here the all-zero embedding, applied only to the tokens of the user's text, so the chat template around them stays fixed while the user's words are what fades. Sundararajan et al. [30] recommend all-zero baseline for text, and two alternatives are available: the end-of-turn token's embedding, and the mean embedding over the vocabulary. Section 4.7 reports all three measured on the same twenty-three prompts, where the all-zero embedding separates content words from function words better than the alternatives, so the recommended baseline is also the one that performs best here. The third setting is a target: Equation 3.1 needs a single number to differentiate, which a model producing a sentence does not supply, so the number used here is the log-probability the model assigns to the recorded answer, summed over its tokens.

Leave-one-out needs something the other two do not: an explicit map from each word to the subtokens that spell it, since deleting a word means removing exactly those positions and closing the sequence up. The faithfulness metrics of Section 4.4 reuse that same deletion path, since they too work by deleting a chosen set of words and re-scoring. Building the map splits the templated prompt into three parts: the chat template before the user's words, the words themselves, and the template tail that follows them. Only the middle part changes. A deletion rebuilds the sequence with the template on either side - left exactly as it was - and one word missing from between them, and what comes back is scored against the same starting point: how likely the model finds the answer when nothing is deleted, computed once at the start.

One thing applies to all three methods. The model does not read words, it reads tokens, which are often smaller than a word. So a name like "Frankenstein" may arrive as three tokens. An explanation has to report on words, because those are what the reader wrote and what the reader will look at, so several token scores have to become one word score somewhere. All three methods cut words at the same places, using the marker the tokenizer puts on the first token of each word. What they cannot share is the rule for combining the scores inside a word. Integrated Gradients keeps the token furthest from zero in either direction, because its scores are signed and a word that argues against the answer has to survive with its sign intact. Attention keeps the largest, which comes to the same thing for weights that leave a softmax and are never negative. Leave-one-out combines nothing at all: it

never scored a token, having deleted a whole word and measured the consequence once. Each rule follows from what its method produces, so the difference is forced rather than chosen. It still means the three explanations do not reach word level by one identical route, and a difference between them could in principle come from that step rather than from the methods themselves.

The three methods each derive their own list of words from the tokenizer, and a comparison between two of them is only meaningful if those lists agree, since a correlation runs over positions and position x has to mean the same word in both. One list therefore has to be authoritative, and the one leave-one-out builds is used, because that method cannot work without it: deleting a word means knowing which token positions spell it, where the other two produce their word lists only as a byproduct of reporting. The scores returned by those two are fitted to it before anything is correlated. In the ordinary case there is nothing to fit, since all three read the same word-start marker and arrive at the same list, and the step exists for the cases that do not, such as unusual punctuation, where a method returning too few scores has the remainder filled with zeros and a warning printed.

Generation is capped at eight tokens, and the cap is set by cost: Integrated Gradients needs six hundred forward and backward passes for every prompt and leave-one-out re-scores once per word, so the length of the explained answer multiplies the most expensive part of the measurement. Eight tokens is enough for the opening clause of a short factual answer, which is a real limitation of what these numbers describe, since they are about how a question shapes the start of an answer.

4.2 Criteria for Evaluating Explanations

There is no formal definition of a good explanation to compare three of them against. What exists instead are criteria that each capture one aspect of an explanation's quality, and three are used in this thesis:

- **Faithfulness**: describes how accurately an explanation represents the model's reasoning [7]. Since it is not directly observable what the model relied on, faithfulness also cannot be measured directly. So the measurement is indirect: the input is perturbed, and the model is checked for changes in the prediction.

- **Robustness**: describes whether an explanation is consistent when the input is changed but its meaning is preserved. Mersha et al. [16] define it as the change in an explanation under perturbed inputs; here the perturbation is

paraphrasing. If two equivalent inputs produce contradictory explanations, at most one of them describes the model, and a reader cannot tell which.

- **Plausibility**: describes how much sense the explanation makes to the user reading it [7], assessed by user studies.

A method can be faithful but not robust, tracking real model behaviour that shifts under a reworded prompt, or robust but consistently wrong. They measure different things, so no method is expected to lead on all three. They are also not answered in the same way. Faithfulness and robustness are properties of the explanations themselves and are measured computationally in this chapter, which is what RQ1 asks. Plausibility is a judgement a reader makes, so it cannot be measured here. It is instead a design target: RQ2 asks how an interface should be built to support it, and the requirements of Section 5.1 answer that from the human-computer interaction literature. A study to assess it is specified in Section 5.5.

A dangerous combination is an explanation that is plausible but unfaithful. Bibal et al. [7] report an experiment by Pruthi et al. [22] that shows how far this can be pushed. Their models were trained on gender-biased data with a loss function that penalised attention on the tokens carrying the bias. The models went on using those tokens, but the tokens no longer appeared in the attention map, so a reader auditing the model for fairness would have concluded that it was unbiased. Such an explanation is worse than none at all, because it supplies exactly the confidence that a missing explanation withholds. A method scoring well on one criterion is therefore never treated as evidence about the other.

4.3 Choosing a Faithfulness Metric

The studies of Chapter 3 disagree about which method is actually faithful, and Chan et al. [8] show that part of that disagreement lives in the measurement itself. Two removal-based metrics, scoring the same four explanations of the same prediction, can rank one of them best and worst. The metric is therefore a choice to argue for, and Chan et al. [8] supply the grounds. They compared six removal-based metrics on two axes, diagnosticity and computational cost. Diagnosticity asks whether a metric prefers a more faithful explanation to a less faithful one. No ground-truth faithful explanation exists to compare against, so randomly drawn importance scores serve as the unfaithful reference, and a metric is diagnostic to the extent that it ranks a real method's explanation above a random one. Computational cost is measured directly, as the number of forward passes a metric needs per explanation.

The six removal-based metrics fall into three groups on the two axes (Figure 4.1). The two decision-flip metrics, which check whether removing the top tokens flips the predicted *label*, are cheap, but neither survives. DFMIT asks only whether removing the single most important token flips the decision, and it reaches 8–10 % diagnosticity, because that flip happens in under 14 % of cases to begin with and the metric therefore scores almost everything alike. DFFOT asks what fraction of the tokens has to be erased before the decision changes, which is diagnostic enough at 64–68 %, but Chan et al. set it aside because the answer depends on the model's bias toward particular classes as much as on the explanation. The two correlation-based metrics, CORR and MONO, match comprehensiveness and sufficiency on diagnosticity (roughly 67–75 % against 74–77 %) but cost roughly seventeen times as many forward passes, since they score one token at a time rather than a fixed grid of five thresholds. That leaves comprehensiveness and sufficiency, COMP and SUFF in the figure. They are among the most diagnostic of the six and the cheapest to run, so the choice needs no trade-off: Chan et al. rank on diagnosticity and break ties on cost, and here the same two metrics win on both. This thesis uses them.

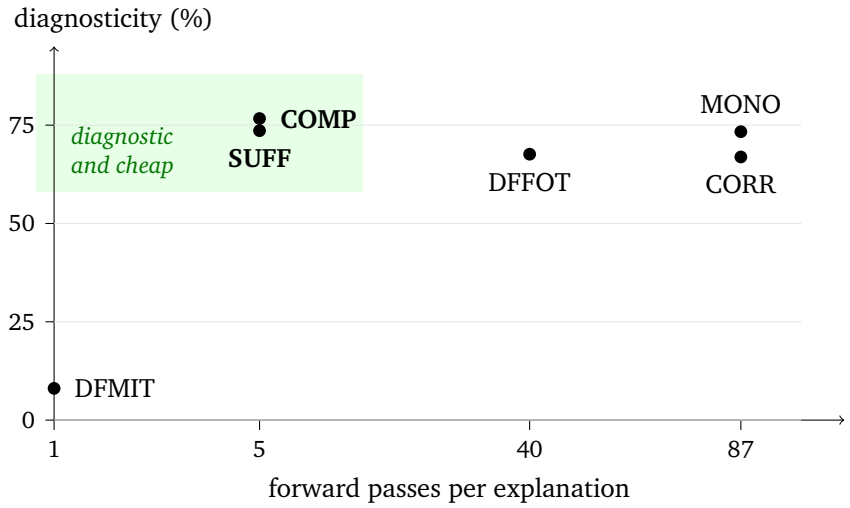


Fig. 4.1: Diagnosticity against computational cost for the six removal-based faithfulness metrics compared by Chan et al. [8], on BERT and averaged over their three datasets. The horizontal axis is logarithmic. Comprehensiveness (COMP) and sufficiency (SUFF) occupy the preferred corner: no other metric is both more diagnostic and cheaper. MONO and CORR pay seventeen times the passes for no gain in diagnosticity, and DFMIT is the cheapest possible metric and the least diagnostic one.

4.4 Evaluation Framework

This section describes how each of the three criteria is turned into a measurement. Every decision below was made before any measurement was run.

The three methods do not report on a single scale, which Chapter 3 noted and which might look like an obstacle to comparing them. But it is not, because every instrument used here is rank-based: the removal metrics work on a fixed fraction of the highest-scoring words, and the agreement and robustness measures are rank correlations. Normalisation by a sum or a maximum is a monotone rescaling, so it cannot change an ordering and cannot change any quantity computed here. The measurements accordingly use the scores as each method produces them, with no rescaling, matching Jain and Wallace [14] and Serrano and Smith [27], who both correlate unrescaled importance values. Their sign is a separate question, settled below.

Faithfulness is measured with the two removal-based metrics that Chan et al. [8] recommend after grading six candidates by diagnosticity and by cost:

- **Comprehensiveness (COMP)**: removes the top-ranked words and measures how far the output falls. A faithful explanation should produce a large COMP.

- **Sufficiency (SUFF)**: keeps only the top-ranked words and measures how far the output falls. A faithful explanation should produce a small SUFF.

Both are averaged over $q \in \{1, 5, 10, 20, 50\}$ percent of the prompt's words, following the grid of DeYoung et al. that Chan et al. [8] adopt unchanged. Three further decisions are fixed here:

- **Ranking** is by the *absolute value* of the score. COMP asks whether removing what an explanation marks as mattering changes the output, and a word that strongly suppresses the answer matters. Ranking by signed value would sort such words to the bottom and treat them as irrelevant.

- **Removal** means deleting from the sequence rather than replacing with a mask symbol, which is the more defensible option for a decoder-only model whose pretraining objective involves no mask token.

- **Granularity** is complete words rather than subword tokens, matching the level at which all three methods report.

COMP and SUFF remove tokens and observe how the output moves, which is the definition of leave-one-out given in Section 3.3. The metric and one of the methods it grades are therefore built on the same operation: leave-one-out ranks words by the drop in output probability caused by removing them, and COMP rewards exactly the explanation whose highly ranked words cause a large drop when removed.

Two things keep this from being fatal. The correspondence is close but not exact, because leave-one-out removes words one at a time while COMP removes the top k together, and the two come apart as soon as words interact: a pair that is individually redundant but jointly necessary earns low leave-one-out scores and still produces a large drop when both are taken out, so the advantage leave-one-out has is real but bounded. The comparison also does not rest on the removal metrics alone, since the rank correlations described below involve no removal at all and therefore favour no method structurally. That second point is what makes the first readable. If leave-one-out leads on COMP while all three methods correlate strongly with one another, the advantage is small in practice. If it leads while correlating weakly, the lead is plausibly an artifact of the metric, not a property of the method.

To establish that the metric distinguishes an explanation from noise, each method is additionally compared against random importance scores drawn uniformly, following Chan et al. [8], who treat a random interpretation as unfaithful. If a real method could not beat random scores on COMP, the metric would not be separating faithful explanations from noise in this setting, and no ranking built on it would carry meaning.

Agreement between methods and robustness within one method are two quantities that involve no removal. Both are measured with Kendall's τ, defined in Section 3.1. Ranks are what the three methods have in common: an attention weight, a sum of gradients and a log-probability difference are not on one scale, so a correlation on raw scores would be reading the scales as much as the explanations. The τ-b variant is used throughout, because attention weights tie frequently on short prompts and τ-a would count those ties as disagreement. Using the same statistic as Jain and Wallace [14] also lets the numbers below be read against theirs. It is applied in two different ways:

- **Agreement**: τ between the raw score vectors of two different methods, computed for each pair on each prompt and averaged.

- **Robustness**: the same statistic within a single method, correlating the explanation of an input against the explanation of a meaning-preserving variant. Variants that change the number of words are recorded as incomparable rather than silently dropped.

Agreement is not a faithfulness metric and is not treated as one. Three methods may correlate closely and be wrong together. What it establishes is whether the methods tell the same story, which is the claim Section 3.4 makes and which nothing else here tests. It also connects this comparison to the studies it follows. Jain and

Wallace [14] report τ between attention and gradient importance from 0.08 to 0.47 across twelve datasets on recurrent encoders, most of them between 0.3 and 0.45, with leave-one-out and gradients correlating strongly with each other but not with attention. Those results came from encoder classifiers over a single scalar output, and Chapter 3 argued that the decoder-only setting differs in what an explanation is anchored to. Applying the same statistic here is what allows Section 4.7 to say whether their finding transfers.

For the gradient–leave-one-out pair, τ is reported both on signed scores and on absolute values. Both methods can produce negative scores here, so a gap between the two figures separates methods that disagree about which tokens matter from methods that agree about which tokens matter but disagree about the direction of their influence. Attention weights cannot be negative (Section 3.2), so every comparison involving attention is necessarily unsigned.

Plausibility, the third criterion, is not measured computationally at all. It concerns whether a person finds an explanation convincing and usable, and it is assessed in the user study of Chapter 5. Plausibility is necessary for an explanation to be usable but does not establish that it is useful, which is why the study observes what participants do with the interface rather than only asking them to rate what they are shown.

4.5 Hypotheses

The comparison motivates three hypotheses:

1. **Attention is the least faithful of the three**, scoring lowest on mean comprehensiveness and highest on mean sufficiency (Section 4.4) across the prompt set. Two results from Chapter 3 support this: Jain and Wallace [14] show that attention weights correlate only weakly with gradient and leave-one-out importance and that alternative weight distributions can leave a prediction unchanged, and Serrano and Smith [27] find that a gradient ranking identifies smaller decision-flipping token sets than attention does in ten of sixteen models. Bastings and Filippova [6] argue the general case that saliency methods are better suited than attention to answering which input a model used, and Sundararajan et al. [30] give a property-based reason to expect Integrated Gradients in particular to outscore it: the method satisfies Completeness, so its attributions sum to the difference the model's output actually moved, which attention weights make no claim to do.

2. **Leave-one-out scores highest on comprehensiveness**, for a structural reason: as set out in Section 4.4, comprehensiveness is itself a removal-based metric, so leave-one-out is assessed partly against its own objective. The prediction is therefore not a strong one, since the metric and the method share an operation and a lead here is close to a restatement of how leave-one-out's scores were computed. It is read alongside the agreement results, which involve no removal and so favour no method structurally.

3. **The three methods disagree substantially**, with mean pairwise Kendall τ below the level at which the explanations could be treated as interchangeable. The attention–gradient pair is expected to agree least, following the results of Jain and Wallace [14] set out in Section 4.4.

4.6 Tasks and Prompts

Besides the hypotheses, the task and the prompts need stating. The model answers one English question at a time from its parameters alone, with no retrieval and no context document. No standard benchmark fits: a benchmark scores answers against correct ones, and what is compared here is not the answer but three explanations of it, for which no ground truth exists. The prompts were therefore written for this study, to vary the property that the three methods are most likely to differ on: how clearly the answer depends on particular words of the question.

Twenty-three prompts are used, each a single sentence, in five groups.

- **Short factual** (6 prompts), where one correct answer exists and the word carrying it is obvious, e.g. *What is the capital of France?*

- **Explanatory** (5 prompts), where the answer is a mechanism rather than a lookup, e.g. *Explain why the sky appears blue.*

- **Opinion and advice** (4 prompts), where no single correct answer exists, e.g. *Is it a good idea to skip breakfast?*

- **Summarization and multi-part** (4 prompts), which ask for long answers, e.g. *Summarize the causes of the French Revolution.*

- **Negation and distractors** (4 prompts), where a word is expected to push *against* the answer, e.g. *Which planet is not a gas giant: Jupiter, Saturn, or Mars?*

The groups are reported separately. An average over twenty-three prompts can hide a method that succeeds on one kind of question and fails on another, and the negation group in particular separates reading a word from depending on it. A model answering *Which planet is not a gas giant* has to read *not*, so attention will weight it highly whether or not it changed the answer; removing the word is what settles that, since the answer flips without it.

Robustness needs a second set of prompts, since it asks whether an explanation survives a rewording of the question it explains. Eighteen pairs are used. Six of the twenty-three prompts, one from each group and two from the short factual one, were each given three variants, and each variant is compared against its base. Six substitute a word that changes the answer (*capital of France* to *capital of Spain*), one for each base prompt. Ten reword the question without changing it (*Who wrote* to *Who authored*). The remaining two change a function word in a way that inverts the question (*is not a gas giant* to *is indeed a gas giant*, *a good idea* to *a bad idea*). Only the second kind is a paraphrase in the strict sense, and only for it should an explanation be expected to stay put. The other two kinds are included because a method that produces the same explanation for *France* and *Spain* is not robust but insensitive, and the criterion of Section 4.2 would not distinguish them without such a case.

4.7 Results

The measurements described in this chapter were run once over the twenty-three prompts and the eighteen paraphrase pairs, with the settings that produced each number recorded alongside it. No prompt failed, and no paraphrase pair was discarded. Generation is greedy throughout, taking the highest-probability token at every step rather than sampling, so the model returns the same answer for the same prompt on every run and each method is a deterministic function of that answer. Re-running the whole measurement on the same machine reproduced every figure below exactly.

Following Chan et al. [8], every prompt was additionally scored against importance values drawn at random, and the question is whether a real method beats that baseline. It does, on both metrics and by a clear margin. Random importance reaches a mean comprehensiveness of $+0.73$, where the three methods lie between $+1.20$ and $+1.81$. On sufficiency, where a lower score is better, random reaches 2.92 against the methods' 1.94 to 2.36. Those are averages, and the per-prompt counts in Table 4.1 are less uniform. Leave-one-out beats the random baseline on all twenty-three prompts and attention on twenty-two, but Integrated Gradients

on only seventeen: on the six of them, deleting an arbitrary set of words damaged the answer as much as deleting the words Integrated Gradients had marked as the important ones. The metric is therefore diagnostic in this setting and a ranking built on it carries meaning, but Integrated Gradients does not reliably clear the bar the baseline sets.

Tab. 4.1: Faithfulness over 23 prompts. Comprehensiveness (COMP) is better when high, sufficiency (SUFF) when low; both are averaged over the removal grid $q \in \{1, 5, 10, 20, 50\}$ percent. The *beats random* column counts prompts on which the method's COMP exceeded that of a random importance vector on the same prompt. Standard deviations are across prompts. Attention is reported at the late layer; Table 4.3 compares the three depths.

Method	COMP ↑	SUFF ↓	beats random
Attention (late layer)	$+1.35 \pm 0.63$	1.94 ± 1.04	22/23 (96 %)
Integrated Gradients	$+1.20 \pm 0.62$	2.36 ± 1.17	17/23 (74 %)
Leave-one-out	$+1.81 \pm 0.99$	2.18 ± 1.19	23/23 (100 %)
Random baseline	$+0.73 \pm 0.59$	2.92 ± 1.18	—

Leave-one-out achieves the highest comprehensiveness among the methods, which Section 4.4 has already given reason to discount: the metric and the method are built on the same operation. The size of the lead is nonetheless worth establishing. Twenty-three prompts is a small set, and an average over them could be driven by a handful of favourable questions. A paired bootstrap tests whether it is. For each prompt the difference between two methods' comprehensiveness is computed, which leaves twenty-three differences. A new set of twenty-three is then built by picking from those differences at random, with each one put back before the next is picked, so that some appear more than once and some not at all. The average of the new set is taken, and the whole procedure is repeated ten thousand times. If the lead survives that, it does not depend on any particular prompt being present. The middle $95\,\%$ of those ten thousand averages is the interval reported below, and taking the differences prompt by prompt is what makes the comparison fair. A hard question drags both methods down, and subtracting one from the other removes the difficulty of the question from the result.

Applied to the comprehensiveness means of Table 4.1, the gap between leave-one-out and Integrated Gradients comes out at 0.61, with an interval of $[0.31, 0.99]$, and between leave-one-out and attention at 0.46, with $[0.22, 0.73]$. Neither interval reaches zero, so both leads hold up. Attention leads Integrated Gradients by 0.15, but its interval runs from -0.10 to $+0.39$, which spans zero. On some resamplings attention came out ahead and on others Integrated Gradients did, so the prompt set is too small to separate the two - which is not the same as finding them equally faithful.

Sufficiency orders the methods differently from comprehensiveness. Attention is best on it at 1.94, leave-one-out second at 2.18, and Integrated Gradients last at 2.36, against the random baseline's 2.92.

Tab. 4.2: Robustness. Kendall's τ_b between a prompt's explanation and the explanation of each of its three variants, over six base prompts and eighteen pairs in total. No pair was discarded.

Method	mean τ_b	sd	min	max
Attention (late layer)	$+0.86$	0.16	$+0.56$	$+1.00$
Leave-one-out	$+0.54$	0.30	-0.20	$+1.00$
Integrated Gradients	$+0.53$	0.36	-0.14	$+1.00$

The ranking inverts on robustness, as Table 4.2 shows. Attention is the most robust of the three at $\tau_b = +0.86$, and its worst single pair still correlates at $+0.56$. Integrated Gradients and leave-one-out are far behind, at $+0.53$ and $+0.54$, and both invert on at least one pair: the words a method called most important for a question become the least important for a rewording of it. Leave-one-out reaches -0.20 on *Who wrote the novel Frankenstein?* against *Who wrote the novel Dracula?*, where substituting the title changes the answer but leaves the structure of the question, and its mean over that prompt's three variants is $+0.07$. Integrated Gradients fails on a different group: over the three variants of *Summarize the causes of the French Revolution.* Its mean correlation is $+0.02$ - no relationship at all between the explanations of near-identical inputs - and its two worst pairs in the whole set are both there. Attention on that same prompt reaches $+0.97$.

So, together with Table 4.1: **no method wins both.** Leave-one-out leads on comprehensiveness and is second on robustness. Attention leads on robustness and on sufficiency but not on comprehensiveness. Integrated Gradients is last on comprehensiveness and last on sufficiency, and on robustness it is level with leave-one-out.

Tab. 4.3: Attention by depth. The early layer is layer 2, the middle layer 8 and the late layer 16 of the model's sixteen. The robustness column is the mean τ_b of Table 4.2, computed per layer.

Layer	COMP ↑	SUFF ↓	beats random	robustness
Early	$+1.13 \pm 0.55$	2.19 ± 0.95	20/23 (87 %)	$+0.88$
Middle	$+1.27 \pm 0.69$	2.08 ± 1.18	18/23 (78 %)	$+0.89$
Late	$+1.35 \pm 0.63$	1.94 ± 1.04	22/23 (96 %)	$+0.86$

Attention was scored separately at an early, a middle and a late layer of the sixteen, and the two criteria behave differently across the three in Table 4.3. On faithfulness the late layer is the better choice: it leads the early layer on comprehensiveness by 0.21, an interval of $[0.04, 0.40]$, and it recovers a better-than-random ranking on twenty-two of twenty-three prompts against the early layer's twenty. The middle

layer sits between them and separates from neither. On robustness the three are indistinguishable, running from $+0.86$ to $+0.89$ with the late layer marginally lowest. Depth therefore buys faithfulness and costs nothing in stability, which is the opposite of what Section 3.1 suggested. The near-uniformity that Abnar and Zuidema [1] report in deeper layers does not prevent those layers from ranking words better here. The two claims are compatible - comprehensiveness reads the order of the scores and not their spread, so weights can flatten while their ranking improves - but the measurement gives no support for preferring an early layer on faithfulness grounds.

Tab. 4.4: Agreement between methods. Mean Kendall's τ_b between two methods' score vectors on the same prompt, averaged over 23 prompts. The unsigned column is reported only for the gradient–leave-one-out pair, since attention weights cannot be negative.

Pair	τ_b (signed)	τ_b ($\lvert\cdot\rvert$)
Attention – Integrated Gradients	$+0.15 \pm 0.25$	—
Attention – Leave-one-out	$+0.27 \pm 0.21$	—
Integrated Gradients – Leave-one-out	$+0.18 \pm 0.28$	$+0.23$

The agreement in Table 4.4 is close to zero in every pair. The strongest, attention and leave-one-out, reaches $\tau_b = +0.27$. The weakest - attention and Integrated Gradients, $+0.15$. For the gradient–leave-one-out pair, correlating absolute scores rather than signed ones moves agreement little, from $+0.18$ to $+0.23$. Their disagreement is therefore about which words matter, not about the direction in which those words push.

These figures sit at the bottom of the range Jain and Wallace [14] report. Their τ between attention and gradient importance runs from 0.08 to 0.47 across twelve datasets on recurrent encoder classifiers, most of them between 0.3 and 0.45. The attention–Integrated Gradients agreement here, $+0.15$, is low against that cluster. Low agreement therefore survives the move to a decoder-only model, but the conclusion they drew from it does not follow here. They read it as evidence against attention, and attention is the method this chapter finds most robust and most sufficient.

The averages hide a systematic difference between kinds of question, and it falls especially on Integrated Gradients. It beats the random baseline on only three of the six short factual prompts and on two of the four negation prompts, while attention and leave-one-out beat it on every prompt in both groups. Short factual questions are the case the prompt set was built to make easy: one correct answer, and one obvious word carrying it. Integrated Gradients fails to beat a random ranking on half of them. The group mean hides this. At $+0.89$ it sits above the group's random mean of $+0.71$, so the average reports a method that works while three of the six

Tab. 4.5: Mean comprehensiveness by question type, with the number of prompts in the group on which each method beat the random baseline. Group sizes sum to the full prompt set.

Group	n	Attention	Gradient	Leave-one-out	Random
Short factual	6	$+1.47$ (6/6)	$+0.89$ (3/6)	$+1.97$ (6/6)	$+0.71$
Explanatory	5	$+1.17$ (5/5)	$+1.31$ (5/5)	$+1.46$ (5/5)	$+0.61$
Opinion	4	$+1.48$ (4/4)	$+1.86$ (4/4)	$+2.04$ (4/4)	$+0.64$
Summarization	4	$+1.37$ (3/4)	$+1.17$ (3/4)	$+2.02$ (4/4)	$+1.01$
Negation	4	$+1.22$ (4/4)	$+0.90$ (2/4)	$+1.56$ (4/4)	$+0.74$

prompts say otherwise. The two negation prompts it loses on are *The capital of France, not Berlin, is what city?* and *Name a mammal that cannot fly*, both of which turn on a single negating word. Leave-one-out is the only method that beats the baseline on every prompt of every group. The random baseline is itself highest on summarization, at $+1.01$: where an answer is long and draws on much of the prompt, removing arbitrary words is already damaging, so the room between a real explanation and an arbitrary one narrows, and results on that group should be read with that in mind.

Section 3.2 took the all-zero baseline on Sundararajan et al. [30]'s recommendation. Their reason is that training gives small norms to unimportant words, so an all-zero vector approximates unimportance. That is a claim about embeddings, and on this model it does not hold. Content words average 0.961 and function words 0.855, a ratio of 1.12 where the claim needs function words to be much smaller. Both sit close to the whole-vocabulary mean of 0.930. Repeating Integrated Gradients over all twenty-three prompts from three baselines - the all-zero embedding, the end-of-turn token's, and the mean over the vocabulary - separates content words from function words by factors of 1.74, 1.63 and 1.17, against 3.37 for leave-one-out on the same prompts. The shipped baseline is the best of the three and all three sit far below that mark, so whatever spreads Integrated Gradients' mass over function words, it is not the choice of baseline. This measurement does not identify what it is.

Against the three predictions of Section 4.5, the measurement returns one failure, one confirmation and one qualified confirmation. **Hypothesis 1 is contradicted.** Attention was predicted to be the least faithful of the three, scoring lowest on comprehensiveness and highest on sufficiency. But it is the second on comprehensiveness and *best* on sufficiency, and it beats the random baseline on twenty-two of twenty-three prompts against Integrated Gradients' seventeen. The predicted ordering against Integrated Gradients in particular runs the other way, and although that gap's bootstrap interval does not exclude zero, nothing in the measurement supports the predicted direction. The prediction was wrong, but the reason behind it may still hold. Attention in deep layers is spread over representations that already

mix many tokens together, so a weight points less clearly at any single word. That problem is real. But Integrated Gradients has a worse one: its mass spreads over function words, which cost it more than depth cost attention. **Hypothesis 2 holds.** Leave-one-out does score highest on comprehensiveness. Comprehensiveness deletes words and measures the drop, which is how leave-one-out ranks them in the first place, so a high score is close to a restatement of the method's own procedure. However, leave-one-out leads on no other criterion, and on robustness it is second, behind attention. **Hypothesis 3 holds**. Pairwise agreement runs from $+0.15$ to $+0.27$, far below the level the hypothesis set. But its specific prediction that the attention–gradient pair would agree least is correct only narrowly, at $+0.15$ against $+0.18$ for gradient–leave-one-out.

5 User Evaluation

5.1 Requirements for Explanation Interfaces

The previous chapter compared the three methods computationally and no method came out best. Attention, integrated gradients and leave-one-out disagree about which words matter, and the measurements give no reason to trust one of them over the others. That is a finding about the methods. It turns into a problem when one of them is put in front of a person, because a user sees only one method at a time. They get a handful of words marked in their question, and nothing on the screen says the other two exist, or that they would have marked different words. The one ranking they are shown arrives as the explanation. Handing them that, when the previous chapter has just shown there is no single answer to hand over, is what the interface has to fix.

Nguyen et al. [20] survey 53 publications on explainable interfaces. Only 17 % involve users for both analysis and evaluation of a system, as human-centred design would require, and fewer than a third involve them in the analysis at all, before any solution is produced. They suggest this follows from XAI being primarily a technical field, where systems are "mostly seen as a prototype to showcase and test algorithmic feasibility". Miller [18] asks what a good explanation is at all, and observes that most work in explainable AI answers from "the researchers' intuition of what constitutes a 'good' explanation" rather than from the philosophy and psychology of how people actually explain. That question has to be answered before an interface can be designed. The literature gives four requirements for the design of explanation interfaces.

An honest interface should show everything it has. That is the natural assumption, but the first requirement goes against it. Reviewing decades of work in philosophy, cognitive psychology and social psychology, Miller [18] finds that people rarely expect a complete cause. They pick one or two out of many and treat those as the explanation. This is a design constraint, because of an effect Miller draws from Tetlock and Boettger's work on conversation [31]: irrelevant detail does not sit harmlessly beside the relevant kind, it *dilutes* it.

The second requirement is about order and depth: what a user sees first, and how much of it at once. Shneiderman [28] states both as one sequence. Overview comes first, then zoom and filter, then details on demand. He offers it as a starting point for design and not as a complete prescription, but it is still the standard structure for presenting data that cannot be shown all at once. For an explanation interface the overview is the answer with its words highlighted, and the detail is the score behind a word, the attention at a particular layer, or the effect of removing that word.

The third requirement follows from what those first two leave fixed. Whoever designs the interface decides the order and the depth in advance. But explanation, for Miller, is social: a transfer of knowledge inside an interaction, shaped by what the explainer assumes the other person already knows. A display that is the same for everyone cannot adjust to any of that, and Miller notes that explainable AI has not truly adopted socially interactive explanation, unlike the earlier systems that explained rule-based models by answering the user's follow-up questions. Fewer than half of the interfaces Nguyen et al. [20] reviewed are interactive at all, and among those with a finished design, 49.1 % walk the user through the explanation in an order the designer chose. They recommend a matrix architecture instead, one that lets a user move between explanations along more than one dimension. Here those dimensions are, for example, the method and the word. A user should be able to hold one word fixed and ask what each method says about it, or hold one method fixed and read across the whole answer.

The last requirement is the one the sources say least about and this thesis needs most. It concerns what the interface says about the methods themselves: which of them disagree, and how much each can be trusted. An explanation that a user finds convincing but that does not describe the model is worse than no explanation at all, because it supplies the confidence that a missing explanation would have withheld. That combination, plausible but unfaithful, is the one Section 4.2 treats as the dangerous one, and Bibal et al. [7] give it a concrete form: an attention map trained to hide the tokens a model in fact relies on looks clean to a reader and "wrongly leads users to believe that the models are unbiased". An interface that shows three methods at once inherits this problem in a sharper form. When the methods agree, that agreement is easy to read as confirmation, although Section 4.2 notes that agreement between two attributions is no evidence that either one is faithful. When they disagree, the interface must not let the user assume that the clearest view is the correct one. Disagreement between methods has to be visible as disagreement, and what has been measured about each method has to be available next to its output.

Together these four requirements describe an interface that shows several explanations of one answer and lets a user move between them. Nguyen et al. [20] find that

showing multiple instances of an explanation for comparison is the most common visual-hierarchy requirement in their corpus, at 30.2 % of surveyed publications, and they call on researchers to build interactive comparing interfaces. Wang et al. [34] observed users drawing on a range of explanation facilities to reason in different ways, and concluded that more work is needed on integrating multiple explanations and not on producing further algorithms.

Meeting the requirements in form is not the same as meeting them in substance. An interface that shows three highlighted copies of an answer and stops there has done the first only, because what is worth seeing is where the methods diverge and by how much, and a display of that kind leaves the user to find it. The design in Section 5.3 is judged against that standard. But first, Section 5.2 looks at how existing tools for inspecting transformer models have handled it.

5.2 Existing Interactive XAI Tools

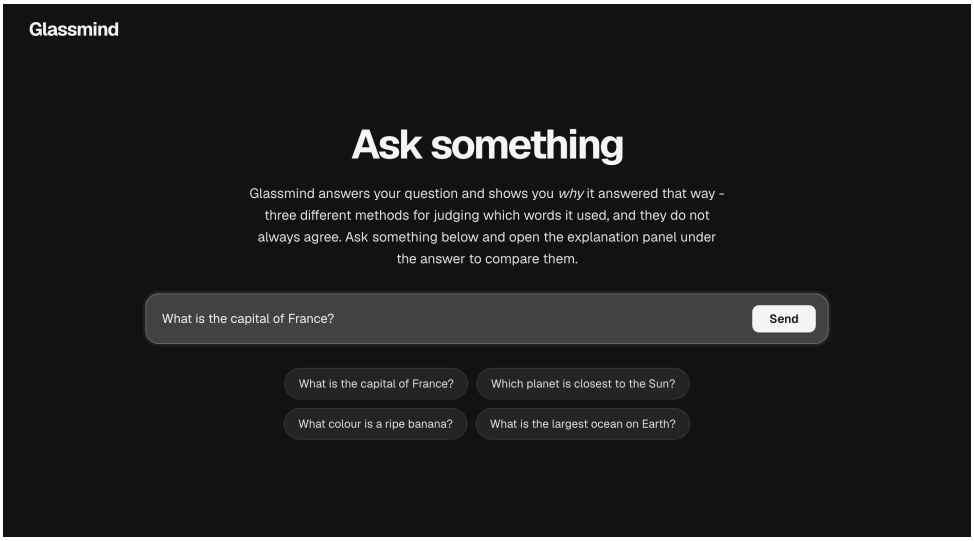
Interactive tools for inspecting transformers, built around attention, already exist. Vig [33] addresses the problem that there is too much to look at: a model with 24 layers and 16 heads produces 384 attention structures for every input, so his tool shows them at three scales, from a single head up to a grid of all of them at once. The tools that followed each add an axis to that picture. Hoover et al. [11] let a user select a token and search an annotated corpus for other tokens the model represents similarly. Wang et al. [37] sets attention against linguistic structure, colouring each head by how far it aligns with syntax or with meaning. Aflalo et al. [2] extend the idea to models that read images and text together, where attention splits into four components and the visual half can be drawn on the image itself.

This thesis takes several things from these works. The token-level heatmap is the standard encoding for attention and is used here as well. Layer selection is an explicit control, as in the model view of Vig [33]: the interface offers an early, a middle and a late layer instead of all of them, for the reason given in Section 3.1. Selecting a word to see what it connects to follows the view of Wang et al. [37].

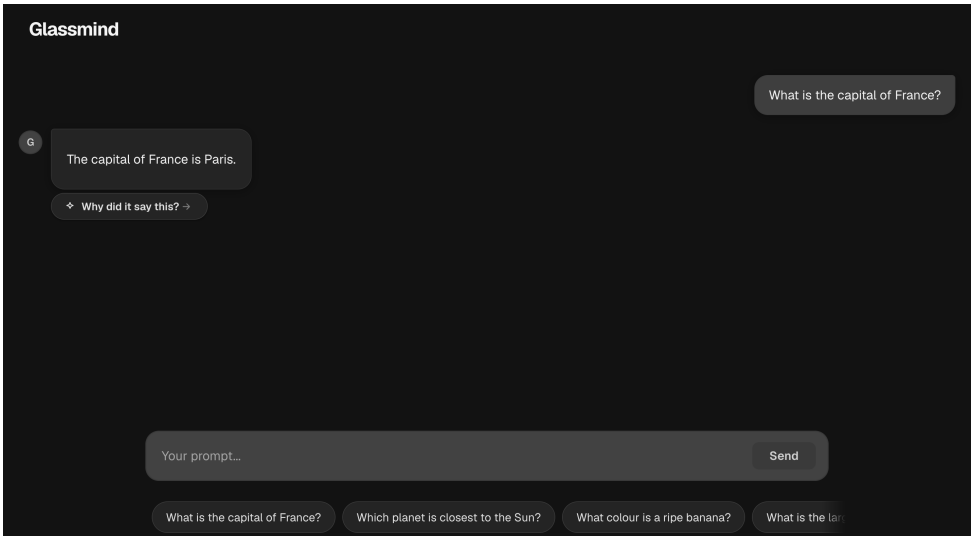
And still, none of these tools show two explanation methods as competing accounts of the same prediction. Each adds an axis of its own - scale, corpus, linguistic structure, a second modality - and each still shows one kind of explanation at a time. The previous chapter found that the three methods disagree about which words matter, and a user given one of them cannot tell. No existing tool sets out to say so. That is what the design in Section 5.3 has to add.

5.3 Interface Design

Users meet the interface as a chat window. They type a question, the model gives an answer, and a button appears under the answer: *Why did it say this?* Nothing is explained until that button is clicked. The first requirement of Section 5.1 is that an explanation shows less than everything, and the strongest form of that is showing nothing until it is wanted. The answer comes first, the explanation is offered, and the user decides whether to see it or not. Figure 5.1 shows both states: the opening screen, and the same screen once an answer has arrived and the explanation is on offer.



(a) The opening screen.



(b) After the answer, with the explanation offered but not yet opened.

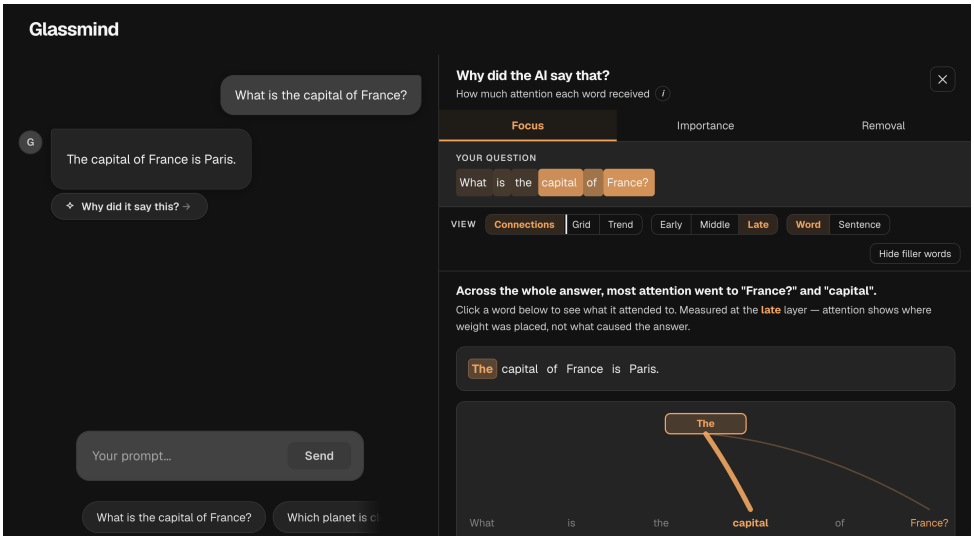
Fig. 5.1: The entry point. The model's answer is shown on its own; the explanation panel opens only when the user asks for it.

Clicking *Why did it say this?* opens a panel, which has three tabs, one for each method of Chapter 3, in the same order as described there: *Focus* for attention, *Importance* for Integrated Gradients, and *Removal* for leave-one-out (Figure 5.2). The tabs carry plain names, because the name of a method says little about what its panel shows to someone meeting it for the first time. An information button on each tab gives a short account of how that method arrives at its numbers, and names it: the tab labels stay plain, but a reader who wants to know that *Removal* is leave-one-out, or that *Importance* is Integrated Gradients, can find it in one click. Naming the methods is what makes the three tabs readable as three different accounts of one answer. Each tab opens with the question, its controls beneath it, then a one-line conclusion and the visualisation under that, with the alternative views in a footer. The panel itself can be resized by dragging its left edge, and the width a user chooses is kept for their next visit, so a grid that arrives too cramped to read is a problem the user can solve. A "hide filler words" switch belongs to the panel as well: it drops words like "is" and "the" from every view at once, and rewrites each tab's conclusion as it goes, since the stated percentages are computed over whatever is currently visible.

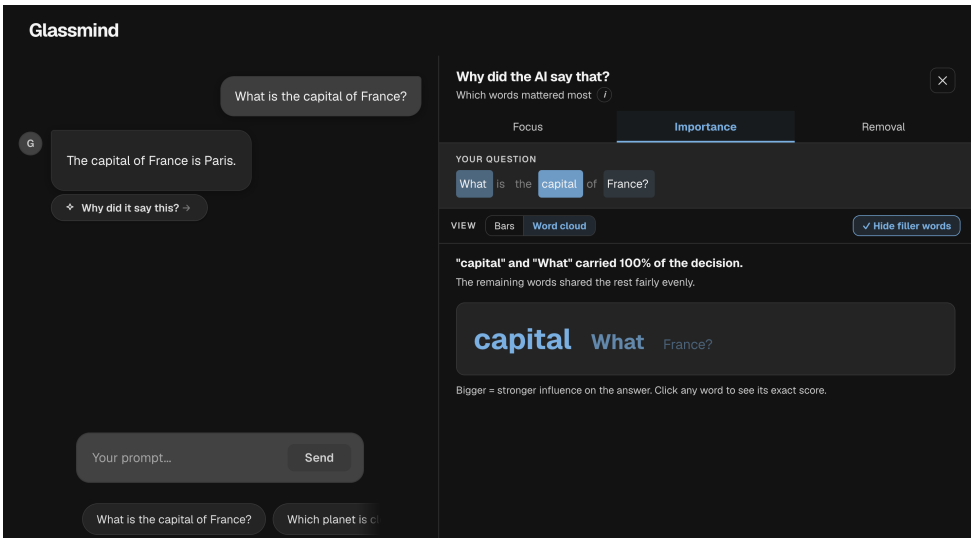
The question always stays fixed. It sits at the top of every tab with its words highlighted, so switching tabs re-highlights the same words with a different method behind them. It is not a new kind of picture, but a second and a third account of the same answer, which are reachable without leaving the panel just by changing between tabs. That is what this interface adds to the tools of Section 5.2.

Focus shows how much attention each word received. Attention is read at three layers, and the panel opens at the early one, where the weights still separate words visibly. That choice is made for legibility, and it is not the most faithful setting. Section 4.7 finds the late layer higher on comprehensiveness than the early one by a margin the paired bootstrap separates from zero, and better at recovering a better-than-random ranking. The default therefore favours the view a reader can act on over the view that scored best, and the layer control sits beside it so that the comparison is one click away. The three sub-views are not three drawings of one number: the connections and grid views are built from one aggregation of the weights and the trend view from another, so they can rank words differently on the same answer. In the first two the user chooses which part of the answer to trace back, one word at a time or a sentence at a time, so the question can be read against the whole answer or against the one clause a user is unsure about. The two are also normalised differently: the word scores are divided by their sum and so add up to one, while each row of the grid is divided by its own largest value to keep the colours legible from row to row. A full-strength cell therefore marks the strongest weight in its row, not a probability of one. Neither scheme reaches any measurement, since Section 4.4 establishes that the comparison runs on ranked raw scores and

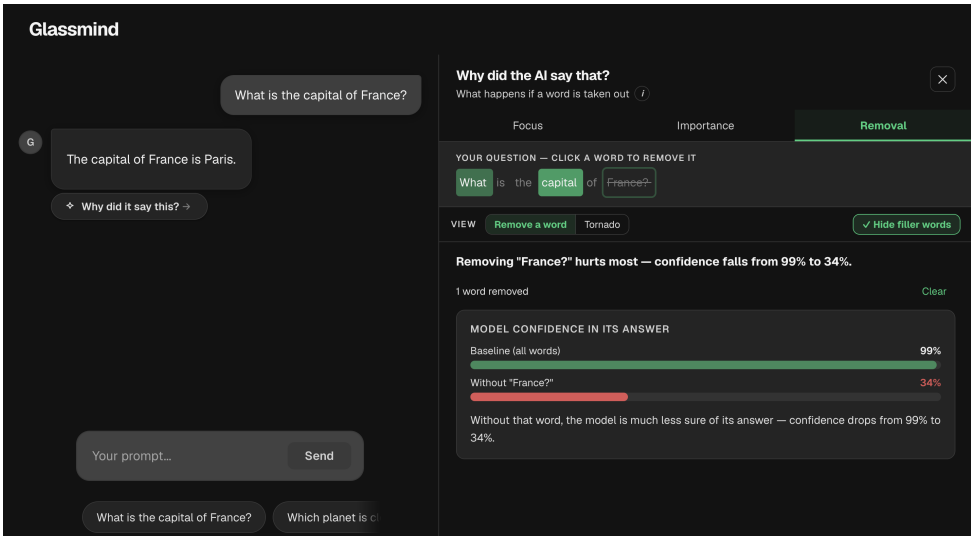
both are monotone rescalings. *Importance* ranks the words of the question by the scores Integrated Gradients assigns them. The panel represents ranking as bars or as a word cloud. The bar view shows the highest-scoring words and the cloud shows all of them. *Removal* uses leave-one-out, as described in Section 3.3. Its first view is interactive. Clicking a word in the question strikes it out, and the panel re-scores the answer without it, one word or several at a time. The second view is a tornado chart, which ranks every word by the size of its effect and gives the effect a sign, so a word whose removal made the model *more* confident is drawn on the other side of the axis. It is the slowest tab, because every word costs another run of the model.



(a) *Focus*: attention weights, read at the late layer. The arcs trace one answer word back to the question words it attended to.



(b) *Importance*: Integrated Gradients, shown here as a word cloud sized by attribution magnitude.



(c) *Removal*: leave-one-out. *France?* has been struck out, and the meter reports the model's confidence with and without it.

Fig. 5.2: The three tabs on the same question. The question stays fixed at the top of each tab, so the highlighting can be compared across methods.

5.4 Implementation of the Interface

Three explanations of one answer means three computations of very different cost sitting behind each panel. This section describes how the system is put together to deliver that.

The server is a FastAPI application built around a single model. It is loaded when the server starts, before the first request, and every route shares that one instance. There is one route per explanation method - `/explain/gradient`, `/explain/attention`, and `/explain/loo`. A single route returning all three would have been simpler to call, but nothing could be shown until the last method had finished. Separate routes let each result be displayed the moment it exists, while the others are still running. A fourth route, `/explain/loo/combo`, re-scores the answer with an arbitrary set of words removed at once. It serves the interactive removal described in Section 5.3 and is the only route driven by a user's selection.

Chat is served by two routes. `/chat` returns a complete answer, and `/chat/stream` returns it progressively: generation runs on a background thread behind a `TextIteratorStreamer`, and the route yields text as the thread produces it. Without this the user would face a blank window for the length of a long answer. Both chat routes build the answer the same way, taking the single most likely word at every step. A model can instead pick randomly among the likely words, which reads more naturally but produces a different answer each time the same question is asked. That is what makes it unusable here: the explanation routes receive only the question and generate the answer again themselves before scoring it, so an answer produced by chance would not be the answer the panel explains, and a user asking why the model said something would be shown the reasons for a sentence they never read. Building the answer the same way everywhere keeps the two identical.

The client is a single component file. React is loaded from a content delivery network and the component is compiled in the browser by Babel, so there is no build step, no bundler, and no toolchain to install - the server hosts the file as a static asset and the page compiles it on load. For a prototype, the cost of that choice is a slower first load, but the benefit is that the interface can be inspected in the same form it is written.

When the user opens the panel, the client requests all three explanations one at a time, starting with the tab the user is looking at and following with the rest in tab order. By the time a user has observed the first panel and come to the third one, the third is usually computed already. Results are cached against the question they explain, so coming back to a panel also costs nothing. Only one request runs at a

time, because the model is a single resident instance and three requests at once would slow down the panel the user is actually looking at. A generation that has started and is already running on the device cannot be called back, so a client that abandoned the response would hide work that carries on regardless. Switching tabs reorders what has not yet begun. The wait is therefore bounded rather than removed. A user who turns to a second method usually finds it waiting, and a user who turns to it early sits through one explanation they did not ask for, not two. The price is that an open panel computes all three explanations, leave-one-out included, which re-runs generation once for every word of the question. Closing the panel stops whatever has not started, so a user who reads one account and moves on is spared the rest. A user who leaves it open pays for three methods to read one.

One important limit is that three methods are not requested with the same generation length. Attention explains up to 150 generated tokens, while Integrated Gradients and leave-one-out explain eight. The reason is cost. Reading attention weights adds one forward pass and does not care how long the answer is, while Integrated Gradients needs six hundred and leave-one-out one per word, so both grow with the length of what they explain. The eight-token cap that Section 4.1 sets for the measurement is kept here for the same two methods. The consequence is that the panels describe different spans of the same answer, and a user comparing them is comparing an account of the whole answer with an account of its opening.

5.5 Proposed User Study

Whether the design works for the people it was built for is a separate question. This section describes the study that would supply the answer to it. The study is proposed and has not been conducted. It is described here as part of the design, not reported as a result.

The proposal is a formative pilot with five to eight participants working through a think-aloud protocol, where each participant says out loud what they are looking at and what they make of it while they use the interface. Surveys of user studies in this field report that experiments comparing conditions *between* participants usually begin at around thirty [26], but the same survey notes that studies which do not split participants across conditions recruit fewer people by construction. Nothing here is split across conditions: one participant uses one interface and may open as many of its tabs as they choose. What a pilot of this size can do is find the places where the interface is misread, which is what a formative study is for and what has to happen before a larger one is worth running. Participants ask their own questions rather than working from a prepared list. The interface is built for someone who wants to

know why they were told something, and a participant answering a question they were handed has no such interest.

A session runs in three parts. The participant first uses the chat alone, with the explanation panel closed, and asks whatever they came to ask until they have an answer they want to examine. They then open the panel on that answer and are asked to say which words they think the model relied on. Only at the end are they told that the three tabs are three different methods and asked whether that changes their reading of what they saw.

The usability standard ISO 9241-11 [12] separates three aspects of use, and each becomes a different observation here. **Effectiveness** concerns whether users achieve their goals. This interface is built on the assumption that a user wants more than one account of an answer, and the simplest way to test that assumption is to watch whether anyone acts on it: having opened the panel, does the participant ever switch to a second tab? Nothing in that observation requires deciding which method is right, which is what makes it usable here. It is worth being clear about what it does not establish. Opening a second tab is not the same as understanding it, and neither is the same as being helped by it. If no participant ever opens a second tab, the multi-method design has failed whatever the individual explanations are worth. If they do, what has been shown is that the interface invites the comparison, but not yet that the comparison does them any good. **Efficiency** weighs what a user gets against what it costs them to get it. Here that cost is the effort of reaching a second opinion: how long a participant takes to get there and how much they have to do. The interface of Section 5.4 spends computation to keep that cost low, working out all three explanations before any of them is asked for. Whether a participant notices the difference, and whether a wait would have stopped them comparing at all, is what the pilot can find out and the implementation cannot. **Satisfaction** is the participant's own view, collected from a questionnaire after the session. The questions move from closed to open and from the concrete to the general, so that a participant is not asked to summarise an experience before being asked about its parts, and they are pretested on one or two users, since a question that turns out to be ambiguous cannot be repaired after the fact.

The study asks people what they think of an explanation, and what they can report is whether it looked convincing. That is plausibility, the criterion Section 4.2 keeps apart from faithfulness, and the two can point in opposite directions. Bibal et al. [7] describe a study in which attention maps were trained to hide the words a model was relying on. The model went on using those words. They simply stopped appearing in the map. A reader sees nothing out of place. Such an explanation would score well on any question this study asks, and it would be wrong. So a participant who is satisfied tells us the interface was convincing, not that it was right.

This is why the second-method observation carries the weight it does. It is not a measure of whether a participant judged correctly. It records whether the interface put them in a position to judge at all. A reader who consults one account has only that account's word for it. A reader who consults two can at least see whether they agree. Whether seeing that changes what anyone believes is the question a larger study would have to answer, and this one is designed to find out whether it is worth asking.

What the pilot is expected to show can be stated in advance. Participants who open the panel are expected to switch to a second tab without being prompted, since the design assumes a single account is unsatisfying. If they do not, the multi-method premise is wrong. The three tabs are expected to be read as three answers to one question rather than three unrelated displays, and any participant who treats them as separate tools is evidence that the interface fails to present them as alternatives. Satisfaction is expected to track how readable a tab is rather than how faithful its method was measured to be in Chapter 4, which is what plausibility diverging from faithfulness would look like from the user's side.

6 Discussion and Conclusion

Two things have been produced. Chapter 4 compared three explanation methods on one model and found no best method: the three disagree about which words matter, and the measurements give no grounds for preferring one. Chapter 5 took that result as a design premise and built an interface that shows all three and lets a user move between them. Both results come with conditions attached. The comparison rests on one model and twenty-three prompts, and the metrics that produced it share an operation with one of the methods they judge. The interface has been built and argued for but never put in front of a user. What follows sets out those bounds, then what would have to be done next, and then what the two chapters together answer.

6.1 Limitations

Every number in Chapter 4 comes from one model. Llama-3.2-1B-Instruct is a one-billion-parameter decoder-only model. It is untested whether the same result would appear on a larger model, on a different family, or on a model trained differently. The prompt set is also small. Twenty-three questions across five groups are enough to show that the methods disagree. But an average over twenty-three items can rest on two or three questions where one method happened to do well, which is why the paired bootstrap of Section 4.7 is used. For two of the three comparisons the intervals hold. For the third, attention against Integrated Gradients, the interval runs from -0.10 to $+0.39$ and the prompt set simply cannot separate them. Robustness rests on even less: six base prompts with three paraphrases each, eighteen pairs in total.

A second limit is inside of the measurements. COMP and SUFF both work by deleting words and watching what happens to the model, which is the operation that creates the out-of-distribution problem described in Section 3.3. So, a high comprehensiveness score can mean either that the removed words mattered, or that the shortened sentence does not look like anything the model was trained on. Leave-one-out also has a problem of its own here, because comprehensiveness removes words and measures the drop, and this is also how leave-one-out works. Leave-one-out therefore scores well on a metric that repeats its own method. That

means the measurement is not neutral between the three methods, and the ranking it produces should be read with that in mind.

The measurements also describe only the beginning of each answer. Scoring stops after eight tokens. Integrated Gradients needs 600 forward passes per prompt, and leave-one-out re-runs the model once for every word of the input, so explaining a longer answer costs more than the comparison could afford. What the numbers describe is, therefore, how the three methods explain the start of a generated answer. The interface, on the other hand, does not keep that cap for all three. Attention is read from a single forward pass regardless of how long the answer is, so the panel lets it explain up to 150 tokens, while Integrated Gradients and leave-one-out stay at eight for the same cost reason. So, a user switching between the tabs is comparing an account of the whole answer with an account of its opening.

Attention was also read at three fixed layers of the sixteen, and as raw weights rather than with the methods that combine information across layers [1]. Where this thesis says attention, it means raw per-layer weights.

The interface of Chapter 5 has not been evaluated with users. Its requirements are derived from the literature in Section 5.1 and the design decisions that follow from them are argued in the text, but whether the result helps to read an explanation is not something that was measured in this work. The study that would answer it is specified in Section 5.5 but was not conducted.

6.2 Future Work

The study described in Section 5.5 is the most direct continuation of this work. Its design is already fixed, down to the number of participants and the protocol. What it counts is whether a participant looks at a second method, not whether they spot a disagreement between them.

Running the comparison on a second model would separate what belongs to Llama-3.2-1B-Instruct from what belongs to decoder-only models in general. A larger prompt set would answer something narrower: the gap between attention and Integrated Gradients is the one comparison twenty-three prompts could not resolve.

The objection raised against both faithfulness metrics in Section 6.1 can be answered. There are ways around deleting words outright. The removed word can be replaced by one that fits the context instead of being dropped [40], each perturbed input can be scored for how far it falls outside the training distribution and down-weighted

accordingly [23], or the model can be retrained on the shortened inputs so that it is scored on data it has actually seen, as described by Wang et al. [36]. Any of these could change what comprehensiveness and sufficiency measure for all three methods.

Other explanation methods could be added to the comparison. Layer-wise relevance propagation adapted for transformers [4] is the strongest candidate on the gradient side. It would only require adapting it to this model. The methods that combine attention across layers [1] are the candidate on the attention side. The late layer measured best in Section 4.7, against the expectation that attention flattens with depth, so a method that aggregates layers instead of choosing one would not have to make that choice at all.

One result of Chapter 4 is left open. Integrated Gradients spreads its scores over function words, and the obvious suspect is the baseline. However, Section 4.7 tried three different baselines, and the scores spread over function words under all of them. So, whatever causes it lies somewhere else. Finding it would say something about how Integrated Gradients behaves on a decoder-only model.

6.3 Conclusion

This thesis compares three families of explanation method on a decoder-only language model and builds an interface that shows all three at once. The comparison covers attention weights, Integrated Gradients and leave-one-out, scored using the same metrics. The interface makes the three explanations of a single answer available side by side.

The comparison led to three main findings. The first is that no method wins on every criterion. Leave-one-out scores highest on comprehensiveness. Attention is the most robust across rewordings of a question and also scores best on sufficiency. Integrated Gradients leads on none of the three. A reader who wants one method has to decide which property matters for their case, because the measurement does not identify a best one. The second is that the three methods disagree with each other almost completely. Pairwise agreement runs from $+0.15$ to $+0.27$, which means two methods looking at the same answer will mostly point at different words. This is consistent with what Jain and Wallace [14] report on recurrent classifiers, so the low agreement they describe also appears in a decoder-only setting. What does not carry over is the conclusion they drew from it. They read low agreement as an argument against attention, and here attention scores best on one of the two faithfulness metrics and beats a random ranking on more prompts than Integrated Gradients

does. The third is about Integrated Gradients. It has the strongest theoretical case of the three, but it performs worst here. It is also last on both faithfulness metrics and fails to beat a random ranking on half of the short factual questions, which are the easiest prompts in the set. Its scores also spread over function words, and the experiment reported in Section 4.7 shows that the baseline is not the cause.

RQ1 asked how the three families differ in faithfulness and robustness. The findings above give the answer: they differ on both, and not in the same order. Leave-one-out leads on comprehensiveness, but part of that lead comes from the metric sharing its procedure, since comprehensiveness also works by deleting words. Attention leads on sufficiency and on robustness, where its worst pair of rewordings still agrees more than the average pair of either other method. Integrated Gradients is last on both faithfulness metrics and level with leave-one-out on robustness. The ranking depends on which criterion is asked about, and that is the answer to the question.

RQ2 asked how multi-method explanations should be presented and how that presentation could be evaluated. The first half is answered by the interface of Chapter 5: requirements taken from the literature on explanation interfaces, a design that follows from them, and a working implementation that shows the three methods on one answer without ranking them for the user. The second half is answered by the study specified in Section 5.5, which fixes the protocol and what counts as a result.

The most important limitation is the narrowness of the evidence. Every number comes from one model and twenty-three prompts, and the robustness figures from eighteen pairs. That is enough to show that the methods disagree and that the criteria order them differently. However, it does not show how they would behave on a larger model, on a different family, or with more prompts. The most important next step follows directly from the other half of the work: the interface exists and the study that would test it is specified, so running that study is what would turn a design argument into an empirical one.

Bibliography

[1] Samira Abnar and Willem Zuidema. „Quantifying Attention Flow in Transformers“. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2020, pp. 4190–4197 (cit. on pp. 16, 17, 43, 58, 59).

[2] Estelle Aflalo, Meng Du, Shao-Yen Tseng, et al. „VL-InterpreT: An Interactive Visualization Tool for Interpreting Vision-Language Transformers“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Demonstrations*. 2022, pp. 21406–21415 (cit. on p. 48).

[3] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, et al. „GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints“. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2023, pp. 4895–4901 (cit. on p. 31).

[4] Ameen Ali, Thomas Schnake, Oliver Eberle, et al. „XAI for Transformers: Better Explanations through Conservative Propagation“. In: *Proceedings of the 39th International Conference on Machine Learning (ICML)*. 2022, pp. 435–451 (cit. on pp. 2, 21–23, 59).

[5] Pepa Atanasova, Oana-Maria Camburu, Christina Lioma, et al. „Faithfulness Tests for Natural Language Explanations“. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL), Volume 2: Short Papers*. 2023, pp. 283–294 (cit. on p. 1).

[6] Jasmijn Bastings and Katja Filippova. „The Elephant in the Interpretability Room: Why Use Attention as Explanation When We Have Saliency Methods?“ In: *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*. 2020, pp. 149–155 (cit. on pp. 2, 38).

[7] Adrien Bibal, Rémi Cardon, David Alfter, et al. „Is Attention Explanation? An Introduction to the Debate“. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*. 2022, pp. 3889–3900 (cit. on pp. 10, 33, 34, 47, 55).

[8] Chun Sik Chan, Huanqi Kong, and Guanqing Liang. „A Comparative Study of Faithfulness Metrics for Model Interpretability Methods“. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*. 2022, pp. 5029–5038 (cit. on pp. 29, 34–37, 40).

[9] Arun Das and Paul Rad. „Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey“. In: *arXiv preprint arXiv:2006.11371* (2020) (cit. on pp. 10, 11).

[10] Ruth C. Fong and Andrea Vedaldi. „Interpretable Explanations of Black Boxes by Meaningful Perturbation“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3429–3437 (cit. on pp. 24–26, 28).

[11] Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. „exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models“. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2020, pp. 187–196 (cit. on p. 48).

[12] International Organization for Standardization. *ISO 9241-11:2018 Ergonomics of Human-System Interaction — Part 11: Usability: Definitions and Concepts*. International Organization for Standardization, 2018 (cit. on p. 55).

[13] Maksims Ivanovs, Roberts Kadikis, and Kaspars Ozols. „Perturbation-based methods for explaining deep neural networks: A survey“. In: *Pattern Recognition Letters* 150 (2021), pp. 228–234 (cit. on pp. 2, 25).

[14] Sarthak Jain and Byron C. Wallace. „Attention is not Explanation“. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 2019, pp. 3543–3556 (cit. on pp. 1, 2, 13–15, 28, 29, 36–39, 43, 59).

[15] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. „Visualizing and Understanding Neural Models in NLP“. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 2016, pp. 681–691 (cit. on pp. 18, 19, 23).

[16] Melkamu Mersha, Khang Lam, Joseph Wood, Ali K. AlShami, and Jugal Kalita. „Explainable Artificial Intelligence: A Survey of Needs, Techniques, Applications, and Future Direction“. In: *Neurocomputing* 599 (2024), p. 128111 (cit. on pp. 1, 9, 10, 33).

[17] Meta AI. *Llama-3.2-1B-Instruct*. `https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct`. 2024 (cit. on p. 30).

[18] Tim Miller. „Explanation in Artificial Intelligence: Insights from the Social Sciences“. In: *Artificial Intelligence* 267 (2019), pp. 1–38 (cit. on pp. 46, 47).

[19] Fuseini Mumuni and Alhassan Mumuni. „Explainable Artificial Intelligence (XAI): From Inherent Explainability to Large Language Models“. In: *arXiv preprint arXiv:2501.09967* (2025) (cit. on pp. 1, 9, 10).

[20] Thu Nguyen, Alessandro Canossa, and Jichen Zhu. *How Human-Centered Explainable AI Interface Are Designed and Evaluated: A Systematic Survey*. 2024. arXiv: `2403.14496 [cs.HC]` (cit. on pp. 9, 46, 47).

[21] Adam Paszke, Sam Gross, Francisco Massa, et al. „PyTorch: An Imperative Style, High-Performance Deep Learning Library“. In: *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. 2019 (cit. on p. 30).

[22] Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C. Lipton. „Learning to Deceive with Attention-Based Explanations“. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2020, pp. 4782–4793 (cit. on p. 34).

[23] Luyu Qiu, Yi Yang, Caleb Chen Cao, et al. „Generating Perturbation-based Explanations with Robustness to Out-of-Distribution Data“. In: *Proceedings of the ACM Web Conference 2022 (WWW '22)*. ACM, 2022 (cit. on pp. 26, 29, 59).

[24]Alec Radford, Jeffrey Wu, Rewon Child, et al. *Language Models Are Unsupervised Multitask Learners*. Tech. rep. OpenAI, 2019 (cit. on pp. 5, 6, 8).

[25]Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. „"Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2016, pp. 1135–1144 (cit. on pp. 9, 28).

[26]Yao Rong, Tobias Leemann, Thai-Trang Nguyen, et al. „Towards Human-Centered Explainable AI: A Survey of User Studies for Model Explanations". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.4 (2024), pp. 2104–2122 (cit. on p. 54).

[27]Sofia Serrano and Noah A. Smith. „Is Attention Interpretable?" In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019, pp. 2931–2951 (cit. on pp. 2, 36, 38).

[28]Ben Shneiderman. „The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations". In: *Proceedings of the 1996 IEEE Symposium on Visual Languages*. 1996, pp. 336–343 (cit. on p. 47).

[29]Ilija Šimić, Eduardo Veas, and Vedran Sabol. „A comprehensive analysis of perturbation methods in explainable AI feature attribution validation for neural time series classifiers". In: *Scientific Reports* 15.26607 (2025) (cit. on p. 27).

[30]Mukund Sundararajan, Ankur Taly, and Qiqi Yan. „Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, pp. 3319–3328 (cit. on pp. 19–21, 23, 28, 32, 38, 44).

[31]Philip E. Tetlock and Richard Boettger. „Accountability: A Social Magnifier of the Dilution Effect". In: *Journal of Personality and Social Psychology* 57.3 (1989) (cit. on p. 46).

[32]Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. „Attention Is All You Need". In: *Advances in Neural Information Processing Systems 30 (NIPS)*. 2017, pp. 5998–6008 (cit. on pp. 5–8, 17).

[33]Jesse Vig. „A Multiscale Visualization of Attention in the Transformer Model". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2019, pp. 37–42 (cit. on p. 48).

[34]Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y. Lim. „Designing Theory-Driven User-Centric Explainable AI". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI)*. 2019, pp. 1–15 (cit. on p. 48).

[35]Junlin Wang, Jens Tuyls, Eric Wallace, and Sameer Singh. „Gradient-based Analysis of NLP Models is Manipulable". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 247–258 (cit. on pp. 20, 24, 29).

[36]Yongjie Wang, Tong Zhang, Xu Guo, and Zhiqi Shen. „Gradient based Feature Attribution in Explainable AI: A Technical Review". In: *arXiv preprint arXiv:2403.10415* (2024) (cit. on pp. 19, 21, 23, 59, 66).

[37]Zijie J. Wang, Robert Turko, and Duen Horng Chau. „Dodrio: Exploring Transformer Models with Interactive Visualization". In: *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*. 2021, pp. 132–141 (cit. on p. 48).

[38] Sarah Wiegreffe and Yuval Pinter. „Attention is not not Explanation“. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 11–20 (cit. on pp. 1, 14, 15, 28).

[39] Thomas Wolf, Lysandre Debut, Victor Sanh, et al. „Transformers: State-of-the-Art Natural Language Processing“. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 38–45 (cit. on p. 30).

[40] Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. „Visualizing Deep Neural Network Decisions: Prediction Difference Analysis“. In: *International Conference on Learning Representations (ICLR)*. 2017 (cit. on pp. 26, 58).

List of Figures

2.1 Axes of explainability and the placement of the three families 11

4.1 Diagnosticity against cost for six faithfulness metrics 35

5.1 The entry point. The model's answer is shown on its own; the explanation panel opens only when the user asks for it. 49

5.2 The three tabs on the same question. The question stays fixed at the top of each tab, so the highlighting can be compared across methods. . 52

List of Tables

2.1 Taxonomic placement of the three method families 11

3.1 Gradient-based attribution methods, ordered by the four classes of Wang et al. [36]. Post-processing is listed after the other classes because it wraps them rather than competing with them. Conservative LRP is listed last because it does not belong to this taxonomy. 23
3.2 What each family measures 28

4.1 Faithfulness over 23 prompts. Comprehensiveness (COMP) is better when high, sufficiency (SUFF) when low; both are averaged over the removal grid $q \in \{1, 5, 10, 20, 50\}$ percent. The *beats random* column counts prompts on which the method's COMP exceeded that of a random importance vector on the same prompt. Standard deviations are across prompts. Attention is reported at the late layer; Table 4.3 compares the three depths. 41
4.2 Robustness. Kendall's τ_b between a prompt's explanation and the explanation of each of its three variants, over six base prompts and eighteen pairs in total. No pair was discarded. 42
4.3 Attention by depth. The early layer is layer 2, the middle layer 8 and the late layer 16 of the model's sixteen. The robustness column is the mean τ_b of Table 4.2, computed per layer. 42
4.4 Agreement between methods. Mean Kendall's τ_b between two methods' score vectors on the same prompt, averaged over 23 prompts. The unsigned column is reported only for the gradient–leave-one-out pair, since attention weights cannot be negative. 43
4.5 Mean comprehensiveness by question type, with the number of prompts in the group on which each method beat the random baseline. Group sizes sum to the full prompt set. 44